

46

LA REVUE QUI N'ARRANGE PAS LE CAS DES DROGUÉS DU FORTH

JUIN 1988

EDITORIAL

Ouf, encore un numéro de tiré! Et nous n'avons pas chomé, car achever les modules 4 et 5, terminer les versions en allemand et anglais de TURBO-Forth, poursuivre la rédaction du manuel, et en même temps s'occuper d'administer l'Association n'est pas une mince affaire. Alors HELP, nous attendons vos articles, tout, n'importe quoi, du plus génial au plus fou en passant par le plus dément, on prend, on diffuse. Qu'avez-vous sur le feu? Rien! Allons, ne racontez pas d'histoire, dites plutôt que vous avez la flemme de fermer une enveloppe et de nous l'expédier. Vous ne voulez rien faire? Pas même un petit coup de pouce? Mais alors, comment que c'est-y qu'on va boucler les numéros suivants? Promis juré, si rien ne vient, je boucle tout en anglais (na!).

Soyons sérieux: un groupe de réflexion est en train de se créer autour du projet F32. Alors, si l'aventure vous tente et que vous regrettez de ne pas avoir participé au départ du dernier PARIS-DAKAR, retrouvez-vous dans la peau d'un JOBS-WOZNIAK en bricolant la puce la plus volcanique de cette fin de siècle. Pour les incultes irréductibles, le JOBS-WOZNIAK est un animal à deux têtes qui se nourrit de pommes, vit son adolescence dans un garage et couve des micro-ordinateurs qui se vendent bien.

SOMMAIRE

Nos coordonnées: ASSOCIATION JEDI 17, rue de la Lancette 75012 PARIS
tel président: (1) 43.40.96.53 ou 3615 SAM*JEDI bal PRESIDENT
tel secrétaire: (1) 49.85.63.67 ou 3615 SAM*JEDI bal SECRETAIRE
FORUM et TELECHARGEMENT de programmes sur 3615 SAM*JEDI
(composer le 33-36.43.15.15 depuis l'étranger)

METACOMPILATEUR

PHENIX

TURBO - FORTH 83

Paramétrable, Homologique, Elliptique, Néoténique, Intégal, eXtra-segment (explications dans le texte)

par Michel ZUPAN

Diffusion: module M5 de TURBO-Forth

PETIT TRAITE DE METACOMPILATION

1) Qu'est-ce qu'un métacompilateur
2) Les vocabulaires de métacompilation
3) La cible
4) Interpréteur et mécanique de métacompilation
5) Les références avant
6) Les en-têtes dans la cible
7) Les types de mots métacompilés
8) Méta-définisseurs de nouveaux types de mots
9) La métacompilation contrôlée
10) Options et utilisation du métacompilateur PHENIX
11) Exemples

1) QU'EST-CE QU'UN METACOMPILATEUR?

Phénix: oiseau mythique qui renait de ses cendres.

La métacompilation est sans doute le summum de la complexité mais aussi de l'intelligence en matière de programmation. Le but de la métacompilation consiste à redéfinir à partir d'un système donné l'intégralité d'un nouveau système. De tous les langages évolués, seul FORTH est suffisamment puissant et maniable pour disposer du concept de métacompilation. Tout système FORTH, sauf peut-être les versions des toutes premières origines du langage, est lui-même conçu et écrit en FORTH. L'outil FORTH d'écriture d'un système FORTH est son métacompilateur et il est devenu d'usage de fournir à l'utilisateur les sources du métacompilateur aussi bien que les sources du langage lui-même métacompilé. Il s'agit-là d'une caractéristique absolument unique pour un langage de programmation.

Pour l'utilisateur, disposer du métacompilateur et des sources du langage signifie qu'il possède avec son outil de programmation d'un SYSGEN extrêmement puissant l'autorisant à redéfinir selon ses besoins propres l'outil-même dont il se sert. Les perspectives offertes sont vertigineuses: le concept illustre à ravir les fondements de l'intelligence créatrice auto-référencée tels que l'a développé par exemple Douglas Hofstadter dans son livre "Gödel, Escher, Bach Les Brins d'une Guirlande Eternelle" (InterEditions 1985).

Historiquement, TURBO-FORTH a été métacompilé en FORTH-83 à l'aide du métacompilateur F83 de Laxen et Perry. Au fil des générations successives, le métacompilateur et le source métacompilé ont été modifiés jusqu'au système actuel qui assure sa propre autogénération intégrale. Rien n'empêche le Forthien de poursuivre cette "boucle étrange" pour son propre compte ou pour créer un FORTH différent.

Nous verrons toutefois que toute entreprise de métacompilation requiert de solides connaissances en FORTH, un QI confort(h)able, beaucoup de rigueur et pas mal de patience dans les débuts pour appréhender le plus complexe de tous les concepts FORTH.

2. LES VOCABULAIRES DE METACOMPILATION

Toute la difficulté de compréhension d'un métacompilateur réside dans l'ambiguïté des mots. Il n'est pas possible d'échapper à cette ambiguïté dans la mesure où le système créé reprend le langage du système créateur. On parle ici de système-hôte et de système-cible. L'ambiguïté est encore accrue du fait qu'un bon métacompilateur est "homologique": il compile "comme" un compilateur usuel, c'est-à-dire que l'écriture de la cible se fait en Forth naturel comme si le code devait être une simple extension de l'hôte. Or la cible, système complet et autonome, ne peut bien entendu pas être un sous-ensemble de l'hôte: elle réside obligatoirement à l'extérieur de celui-ci.

Pour lever l'ambiguïté des mots, FORTH utilise son concept de VOCABULAIRES. Un même mot aura une signification différente selon qu'il appartient à tel ou tel vocabulaire. C'est le CONTEXT, c'est-à-dire la liste des vocabulaires actifs à un instant donné, qui permet au système hôte de savoir quel est de tous les homonymes le sens du mot à interpréter. La gestion des vocabulaires en métacompilation est une étape-clé qui conditionne tout le processus.

Il n'y a pas moins de 8 vocabulaires de travail pour l'hôte métacompilant sans compter les images des vocabulaires de la cible.

ROOT est le vocabulaire minimal du langage. Il sert essentiellement à gérer l'utilisation des autres vocabulaires dans le contexte à l'aide du concept ONLY ALSO.

FORTH est le vocabulaire usuel de l'hôte. Il sert à définir toutes les opérations d'accès mémoire, d'interprétation, de compilation, de métacompilation. La plupart de ses mots auront bien sûr des homonymes dans la cible puisque c'est le but de la métacompilation.

ASSEMBLER est l'assembleur du métacompilateur. Il s'agit le plus souvent de l'assembleur usuel auquel on aura simplement revectorisé les primitives d'accès mémoire pour que l'assemblage se fasse dans la zone d'écriture de la cible. Il peut aussi s'agir d'un cross-assembleur si le processeur de la cible n'est pas celui de l'hôte.

META est le vocabulaire-maître de la métacompilation. C'est lui qui contient tous les mots d'interprétation du métacompilateur. Il s'enrichit en outre au cours de la métacompilation d'un certain nombre de références de la cible sous forme de labels, de constantes, d'adresses de variables dont il peut avoir besoin par la suite.

TARGET est le vocabulaire de la cible. Il faut bien comprendre que TARGET n'est pas le chainage des mots tels qu'ils sont compilés dans la cible: c'est un vocabulaire de l'hôte qui conserve seulement les adresses de ces mots dans la cible. C'est donc plutôt l'image de la cible ou si vous voulez une table des références de la cible. Tous les mots de TARGET ont la même structure: à l'exécution, ils compilent dans la cible le code exécutif d'un mot de la cible déjà métacompilé.

FORWARD est le vocabulaire des références avant. C'est un vocabulaire en tout point comparable à TARGET mais dont les mots correspondent à des références non encore définies dans la cible. A l'exécution ces mots forment une chaine d'adresses où doit être compilée une référence jusqu'au moment où cette référence est définie ce qui permet de "résoudre" cette chaine d'adresses laissées en suspens. Il est évident que la métacompilation doit s'achever avec un vocabulaire FORWARD entièrement résolu sous peine de laisser des "blancs" dans le code de la cible.

TRANSITION est le vocabulaire de structure. C'est lui qui contient les mots d'interprétation immédiate des structures de contrôle comme IF ELSE THEN lorsque le métacompilateur est lui-même en état de compilation de haut niveau dans la cible. C'est META qui décide d'activer TRANSITION quand il métacompile des définitions dans la cible.

USER est le vocabulaire de métacompilation des mots de type USER. Exactement comme le vocabulaire USER de FORTH

contient des homonymes de FORTH pour définir des mots d'usager (essentiellement pour le multitâche), ce deuxième vocabulaire USER contient des homonymes de META pour définir dans la cible le même type de mots.

Il importe que le métacompilateur dispose de mots permettant de bien établir à chaque instant les vocabulaires de contexte ainsi que le vocabulaire courant où sont apportées les nouveaux mots. Il n'est pas inutile même de prévoir des synonymes dans META dans la mesure où celui-ci va rapidement se charger avec des références portant des noms de vocabulaires de la cible. Ainsi le mot FORTH dans META ne va plus placer FORTH dans CONTEXT mais référencer le vocabulaire FORTH de la cible: il faudra utiliser le synonyme immédiat [FORTH] pour parler du vocabulaire FORTH de l'hôte.

Les mots de META comme IN-META, IN-TRANSITION, IN-TARGET ou SWITCH (qui conserve et restitue un contexte), définis en début du métacompilateur sont fondamentaux: c'est leur oubli ou mauvais usages qui sont responsables de tant d'erreurs ou incompréhensions de métacompilation.

3. LA CIBLE

Le système-cible est toujours inerte pour le système-hôte. Pour celui-ci, il ne s'agit que d'une suite d'adresses où est compilé un code qui lui est étranger. En fin de métacompilation, cette zone sera sauvegardée et l'utilisateur pourra sortir du système hôte pour lancer le nouveau système. Ce lancement peut toutefois être réalisé par l'hôte s'il possède un intégrateur. Nous verrons que TURBO-FORTH se génère ainsi en deux étapes: métacompilation du KERNEL puis lancement de celui-ci qui s'étend lui-même avec le NOYAU.

Mais revenons à la cible. Son site peut être très variable selon les systèmes. Ce pourra être une zone non occupée du système hôte, par exemple en mémoire haute entre dictionnaire et tampons. Ce peut être en mémoire virtuelle dans des tampons régulièrement sauvés dans la mémoire de masse. Sur un système disposant d'une grande mémoire multi-programmes, le plus logique est d'utiliser l'espace mémoire hors-hôte.

Dans le cas du TURBO-FORTH, la métacompilation s'opère dans un extra-segment alloué. L'avantage d'un tel système est de pouvoir utiliser des adresses 16 bits réelles, sans aucun calcul de déplacement entre les adresses de métacompilation et les adresses définitives du système créé.

Quelque soit le site de métacompilation retenu, le métacompilateur doit disposer en premier lieu d'une série de mots très simples permettant de lire ou d'écrire dans ce site ainsi que d'un pointeur avançant au fil de la métacompilation dans cet espace. Par analogie avec les mots fondamentaux du compilateur forth @ C@ ! C! DP HERE ALLOT , C, le métacompilateur dispose des mots @-T C@-T !-T C!-T DP-T HERE-T ALLOT-T ,-T C,-T etc qui lui suffisent à définir par la suite toutes les opérations sur la cible. Cette portion du métacompilateur est spécifique: son changement permet d'envisager tout autre site interne ou externe de métacompilation.

Ces mêmes mots d'accès mémoire au site temporaire de construction de la cible seront en outre utilisés pour obliger l'assembleur à n'assembler du code que dans cet espace objet réservé.

4. INTERPRETEUR ET MECANIQUE DE METACOMPILATION

Tout compilateur commence par un interpréteur: un texte source est interprété et les mots traduits en ordres de rangement de certaines valeurs à certaines adresses. Le compilateur usuel compile de nouvelles définitions dans le système FORTH, le métacompilateur compile de nouvelles définitions dans un système cible.

En métacompilation, l'interpréteur de l'hôte reste l'interpréteur naturel du FORTH: les notions comme SOURCE, INTERPRET, INCLUDE ne sont pas modifiées. Le texte source à métacompiler est interprété mot-à-mot en une seule passe. Les mots sont exécutés après recherche dans le ou les vocabulaires de contexte. Bien souvent cependant, l'interpréteur est obligé de faire de courts retour en arrière dans le source pour à la fois métacompiler et conserver les références.

Prenons un exemple qui expliquera bien toute la mécanique de la métacompilation. Supposons que nous soyons en métacompilation META IN-META et que nous désirions métacompiler une variable. Nous écrirons dans le texte:

VARIABLE TEST

Le mot VARIABLE de META va en réalité faire trois compilations :

1) Un en-tête (si les en-têtes sont métacompilés) va être créé avec TEST dans la cible. Le vfa est initialisé, le lfa pointe sur une définition précédente dans le méta-vocabulaire courant pour permettre à l'interpréteur futur de retrouver TEST, et un nfa est compilé avec le nom TEST. Puis un cfa est métacompilé avec le code exécutif d'une variable. Enfin le contenu de la variable TEST est initialisé à zéro.

2) Un mot TEST est également créé dans TARGET. Ce mot possède dans son pfa l'adresse du cfa-cible de TEST. Lorsque le métacompilateur trouvera dans une définition de haut niveau le mot TEST, le TEST de TARGET compilera dans la cible cette adresse "comme si" l'interpréteur avait cherché ce mot directement dans la cible.

3) Mais le mot TEST va encore une fois être créé, cette fois dans le vocabulaire META. Ce troisième mot TEST est une constante contenant l'adresse du contenu (pfa-cible) de la variable TEST. En effet, il va peut-être falloir que le métacompilateur, en exécution cette fois, puisse initialiser la variable TEST. Ce sera possible en écrivant simplement 37 TEST !-T. Ici c'est ce troisième mot TEST qui est interprété par META.

Cette triple compilation suppose évidemment une triple interprétation du mot TEST après VARIABLE avec deux retours en arrière devant TEST dans le flot d'interprétation. Ainsi s'explique la mécanique particulière des mots HEADER TARGET-CREATE ou RECREATE qui compilent une fois, deux fois, trois fois le même mot à des endroits différents.

Toutes les définitions ne sont pas compilées trois fois. Un LABEL par exemple n'est compilé qu'une fois dans ASSEMBLER: c'est une étiquette qui n'a pas de référence dans la cible et n'est utilisée que par le méta-assembleur. Un mot codé ou en haut niveau n'est compilé que dans la cible et dans TARGET: la seule référence de son cfa est suffisante au métacompilateur.

5. LES REFERENCES AVANT

D'ordinaire, pour limiter les risques d'erreur le FORTH refuse de reconnaître un mot qui n'a pas été préalablement défini. En métacompilation, il est très difficile de concevoir un système où tous les mots n'utilisent que ses prédécesseurs. Bien souvent il arrive qu'un mot contienne la référence d'un mot ultérieur. C'est pourquoi, le métacompilateur dispose d'un système particulier de définition et de résolution des références avant.

Soit à métacompiler la définition du mot EXEMPLE contenant le mot ESSAI qui ne sera défini que plus tard. Le métacompilateur va définir alors une référence ESSAI dans le vocabulaire FORWARD. Ce mot pointe sur la dernière adresse cible où doit être placée la vraie référence de ESSAI. Lors de la métacompilation de EXEMPLE, ESSAI placera une adresse-cible chaînant sur la précédente adresse à référencer tandis qu'ESSAI pointera sur cette adresse-cible dans EXEMPLE. Ainsi se construit à chaque appel de ESSAI avant sa véritable définition une chaîne d'adresses-cible à référencer. Lorsque ESSAI sera défini, la véritable référence à métacompiler sera replacée de proche en proche dans la chaîne des adresses-cible pointées par le FORWARD ESSAI. Cette opération s'appelle la résolution de la référence avant. Bien entendu la métacompilation complète suppose que la référence avant

ESSAI soit résolue tôt ou tard. En fin de métacompilation il est d'ailleurs prévu que le système affiche la liste des références avant non résolues si le programmeur en a malheureusement oublié dans son source.

Le système de métacompilation des références avant peut être manuel: le programmeur donne l'ordre de créer une référence avant par FORWARD: en préambule de toute utilisation et plus tard ordonne la résolution par RESOLVES quand le mot référencé est métacompilé un peu à l'image de ce qu'on fait habituellement avec DEFER puis IS. Ce système peut être semi-automatique comme dans le métacompilateur F83: si le métacompilateur rencontre un mot inconnu, il crée de lui-même une référence FORWARD. La résolution, en général en fin de métacompilation, reste à la charge du programmeur par l'utilisation du mot RESOLVES.

Le métacompilateur PHENIX est "Elliptique": il possède un système de références avant entièrement automatique. Si un mot à référencer est inconnu, il est aussitôt créé dans FORWARD. La résolution est également automatique: à chaque définition d'une nouvelle référence, le métacompilateur cherche à résoudre le même mot dans FORWARD. Le programmeur n'a plus à se préoccuper des procédures finales de résolution: il lui suffit de définir tous les mots dont il aura besoin dans n'importe quel ordre.

6. LES EN-TETES DANS LA CIBLE

Si le système cible ne doit pas posséder d'interpréteur ou de compilateur ce qui est notamment le cas d'un programme ou module d'application, il est économique de métacompiler la cible sans en-têtes c'est-à-dire sans vfa, lfa et nfa (champs vue, lien et nom) pour ne garder que les cfa et pfa (champs code exécutif et paramètres). Il suffit pour ce faire de donner à la variable META WIDTH la valeur zéro. Cette variable contient en fait la longueur maximale d'un nom de mot: elle est d'ordinaire fixée à 31 (codage sur 5 bits) en FORTH.

Un système cible métacompilé sans en-tête est un objet FORTH totalement fermé: il ne contient que du code et des adresses de code appelées par l'interpréteur interne du langage. Il est impossible de le décompiler, un DUMP ne permet aucun repérage, le désassemblage est inefficace pour en saisir le fonctionnement. Outre le gain en place, un tel système offre donc une excellente protection du travail conceptuel de son auteur. Les grandes applications commerciales écrites en FORTH sont bien évidemment compilées sans en-têtes.

Dans un système ouvert, il peut être intéressant de métacompiler certains mots sans en-têtes: des primitives qui deviendront inutiles, des protections, des mots transitoires ultérieurement "patchés", des mots cachés. Attention cependant aux risques de plantages si ce système permet l'utilisation d'un décompilateur.

7. LES TYPES DE MOTS METACOMPILABLES

Le métacompilateur PHENIX TURBO-FORTH offre l'avantage d'être intégral: tous les types de mots utilisés par le système FORTH sont prévus dans le métacompilateur sans qu'il soit nécessaire de définir d'autres méta-définisseurs dans le source contrairement à celui du F83 où le KERNEL contient encore des définitions pour META. Ceci est possible grâce à l'utilisation plus large des références FORWARD.

Voici les 12 types de mots métacompilables:

a) LABEL <mot> ---
un label est une zone codée sans en-tête dans la cible servant de référence pour le méta-assembleur. La référence est créée uniquement dans ASSEMBLER de META.

b) CODE <mot> ---
métacompile un mot de bas niveau avec le méta-assembleur. La référence est créée dans TARGET.

c) VARIABLE <mot> ---
métacompile une variable. Une référence est créée dans TARGET ainsi qu'une constante du pfa-cible dans META.

d) CREATE <mot> ---
métacompile un en-tête et un code exécutif de type VARIABLE mais sans pfa. Double référence dans TARGET et META comme VARIABLE.

e) CONSTANT <mot> n---
métacompile une constante explicite. Une référence est créée dans TARGET et la constante est également créée dans META.

f) 2VARIABLE <mot> ---
comme VARIABLE pour une variable 32 bits.

g) 2CONSTANT <mot> d---
comme CONSTANT pour une constante 32 bits avec référence dans TARGET et double-constante dans META.

h) STRING <mot> n---
métacompile une variable-chaine de longueur maximale explicite. Une référence est créée dans TARGET et la même variable-chaine est créée dans META.

i) DEFER <mot> ---
métacompile un mot différé. Une référence est créée dans TARGET. Le mot IS de META permet de vectoriser naturellement ce mot différé.

j) USER VARIABLE <mot> ---
métacompile une variable USER tâche-dépendante. Une référence est créée dans TARGET et une constante dans META pointant directement dans la table USER de base de la cible.

k) USER DEFER <mot> ---
métacompile un mot différé USER tâche-dépendant. Une référence est créée dans TARGET ainsi qu'une constante dans META pointant dans la table USER de la cible pour permettre la vectorisation de ce mot différé.

l) VOCABULARY <mot> ---
métacompile un vocabulaire dans la cible. Une référence est créée dans TARGET ainsi qu'un pseudo-vocabulaire dans META dont l'exécution modifie le méta-contexte de la cible et prépare l'action du mot META DEFINITIONS qui établit le méta-vocabulaire courant de chainage des définitions dans la cible. Il est très important de considérer que pour l'hôte, il n'y a qu'un seul vocabulaire de références TARGET. Il est donc inutile de préciser dans une métadéfinition le ou les vocabulaires de méta-contexte.

En contrepartie, il est recommandé d'user des homonymes avec une grande prudence en respectant l'historique des métacompilations. Ceci est notamment le cas pour les mots du métavocabulaire USER (tout particulièrement CREATE).

8. META-DEFINISSEURS DE NOUVEAUX TYPES DE MOTS

Il peut être nécessaire à l'utilisateur de définir d'autres types de mots que ceux prévus dans le FORTH usuel puisque c'est une des caractéristiques du FORTH d'autoriser justement la définition de nouveaux mots de définition. Citons par exemple les tableaux à une ou plusieurs dimensions, le type fichier, les vecteurs multiples d'exécution etc...

Dans ce cas, il faudra définir pour et dans META, en cours de métacompilation du source, les mots de métadéfinition adéquats. Le processus est assez délicat bien-sûr mais une bonne compréhension des mécanismes de métacompilation et l'exemple des douze types prédéfinis permet de métacompiler les plus complexes structures.

Tout d'abord, il faut définir dans FORWARD, avec FORWARD: <TYPE>, le code exécutif des mots de deuxième génération. Ensuite il faut définir dans META (IN-META), le métadéfinisseur. Utiliser alors H: et H; pour définir ce mot puisque : et ; sont réservés à la métacompilation

elle-même. Dans cette définition, on placera l'ordre de métacompilation du cfa du type spécial par [FORWARD] <TYPE>. Si le mot méta-défini doit subir une initialisation quelconque en fin de métacompilation, prévoir une recréation (RECREATE) dans META. Si ce mot est susceptible d'être exécuté pour modifier la métacompilation elle-même, le recréé de META doit agir en conséquence. C'est le cas des pseudo-vocabulaires recréés par VOCABULARY. Lors de cette étape, bien faire attention à compiler dans META les bons mots du système hôte (utiliser [FORTH] et [META] dans la définition H:).

Ensuite, il faudra métacompiler le mot de définition et résoudre ainsi le code exécutif des mots métadéfinis. S'il s'agit d'un mot de définition bas niveau, un LABEL <TYPE> code et une définition :...;USES <TYPE> , ou une définition :...;CODE LABEL <TYPE> code résoudra toute la structure du nouveau type. S'il s'agit d'un mot de très haut niveau, c'est-à-dire d'une structure CREATE...DOES>, il faudra faire suivre le DOES> de META par la référence <TYPE> de FORWARD. Là encore, le méta-compilateur PHENIX permet une résolution immédiate.

9. LA METACOMPILATION CONTROLEE

Le métacompilateur PHENIX du TURBO-FORTH introduit un nouveau concept original: la métacompilation contrôlée d'applications compactes. Le terme "Néoténique" est un terme zoologique qualifiant un organisme capable de se reproduire sous forme larvaire. Mais il convient tout d'abord de rappeler ce qu'est un COMPACT.

Un COMPACT est une application FORTH compilée sans en-têtes ni librairie. Seuls sont compilés les mots nécessaires à l'application définitive. Un tel programme offre évidemment des avantages multiples: gain de place, chargement rapide, exécution optimisée, protection maximale, concept modulaire dans un environnement intégré.

Pour compiler un COMPACT, il convient de métacompiler sans en-tête un système FORTH d'abord limité aux primitives necessaires à l'application puis étendu à l'application complète elle-même. Un utilitaire que l'on appelle Scrutateur Récursif de Primitives permet par une décompilation récursive de connaitre quels sont les mots utilisés dans un mot ou un ensemble de mots d'une application. Le principe en est le "marquage" des primitives au niveau de leurs view- fields.

Plutôt que d'obliger le programmeur de COMPACT à éditer un source limité aux primitives de son application, il est plus pratique de réutiliser le KERNEL propre de TURBO-FORTH avec un système de contrôle asservi au système hôte préalablement indexé. Dans cette option, le métacompilateur PHENIX ne conservera du KERNEL que les mots homonymes "marqués" dans le vocabulaire FORTH de l'hôte. Dans ce mode contrôlé, chaque métacompilation est soit validée dans la cible soit immédiatement oubliée par un retour en arrière. Les références TARGET et FORWARD sont toutefois conservées avec résolution toujours automatique des références valides. Les initialisations portant sur des données non valides n'ont pas d'effet.

Pour créer de façon rapide un COMPACT, il faut donc:

a) compiler une première fois l'application

b) marquer tous les mots dans FORTH nécessaires à l'application à l'aide d'un SRP. Marquer également les vecteurs que l'on désire prendre pour les mots vectorisés (utiliser plutôt des primitives rapides).

c) charger le métacompilateur PHENIX. Donner alors les options sans en-têtes (WIDTH OFF), métacompilation contrôlée asservie à l'hôte (CHECKING ON), sauvegarde en attente d'une suite (SAVING OFF).

d) métacompiler l'ensemble du KERNEL

e) compiler à nouveau l'application sous META cette fois, c'est-à-dire la métacompiler à la suite du KERNEL.

f) terminer la métacompilation END-META après initialisations, option de sauvegarde définitive et nom du fichier de sauvegarde.

Un petit exemple de métacompilation d'un COMPACT accompagne le métacompilateur PHENIX. La réalisation de COMPACTS y apparaît comme un exercice FORTH somme toute assez simple!

Quelques précisions sur le système de contrôle. Le seul vocabulaire détenteur des marques est le vocabulaire FORTH de l'hôte. Si l'application utilise des mots non compilés dans FORTH, alors que dans le COMPACT il n'y aura bien-sûr aucune notion de vocabulaires puisqu'il n'y aura ni en-têtes ni interpréteur, il faudra soit simplifier l'application sur ce point, soit définir dans FORTH des mots fictifs homonymes et individuellement marqués, soit encore supprimer le mode contrôlé lors de la méta-compilation de l'application. Ainsi, même si tous les mots FORTH sont marqués, la métacompilation de KERNEL en CHECKING ON ne donnera pas un TURBO-FORTH complet (exemple le END-CODE du vocabulaire ASSEMBLER)

Le système de compilation contrôlée peut être suspendu avec la directive FORCED. Après cet ordre de META, toute métacompilation est forcée dans la cible sans contrôle d'asservissement. Le contrôle peut ensuite être rendu par la directive UNFORCED.

Les LABELS sont toujours en compilation forcée, même ceux qui peuvent être inutiles au COMPACT, ceci pour éviter tout oubli ou erreur difficile à détecter dans la mesure où les labels ne sont pas marquables et peuvent être utilisés très à distance de leur définition. De même, pour simplifier le marquage et limiter les erreurs, un petit nombre de mots est d'emblée forcé dans KERNEL. Il s'agit de COLD, BYE, de la zone USER avec UP et de 'TIB qui comme UP est initialisé par la procédure codée de démarrage à froid. Dans ces conditions, un COMPACT minimal, qui ne ferait absolument rien, en plaçant juste BYE dans COLD occuperait 340 octets. C'est là le proton du noyau TURBO-FORTH.

La compilation contrôlée peut aussi s'effectuer avec des en-têtes si l'utilisateur désire métacompiler un système avec interpréteur voire compilateur simplifié.

Un système de contrôle différent de celui basé sur l'asservissement au FORTH hôte via les vfa marqués peut être installé en changeant le vecteur du seul mot ?VALID. Ce mot est celui qui contrôle la validité de chaque métacompilation en affectant la variable VALID.

10. OPTIONS ET UTILISATION DU METACOMPILATEUR PHENIX

Le métacompilateur PHENIX propose 4 options modifiables par l'utilisateur. Ces options sont fixées par défaut dans l'optique du clonage in extenso de TURBO-FORTH lui-même.

L'opération de méta-reproduction de TURBO se réalise entièrement sous TURBO-FORTH par la commande INCLUDE META-BAT.

Dans le détail, cette génération s'opère en deux temps. Tout d'abord le métacompilateur produit un KERNEL complet dans le fichier TEST.COM. Puis l'intégrateur lance TEST.COM avec la commande de s'auto-étendre avec le NOYAU. Le noyau comprend les utilitaires usuels de TURBO-FORTH.

Ce procédé offre plusieurs avantages. La compilation du noyau est plus rapide que sa métacompilation. Le noyau comporte plusieurs mots de définitions dont une quinzaine pour le seul assembleur qu'il aurait fallu prédéfinir dans le métacompilateur. Le noyau n'offre guère d'utilité dans un système compact. Il est plus facile d'étendre TURBO en compilation qu'en métacompilation. La compilation autorise la gestion de plusieurs vocabulaires.

En fin d'autogénération du NOYAU, l'utilisateur pourra indexer ses vocabulaires avec les fichiers d'auto-documentation par la commande LEARN puis sauver sa version totale de TURBO par un SAVE-SYSTEM. Avant de quitter ce TURBO-FORTH cible par BYE ou F8, vérifier que le répertoire courant retrouvera bien META-BAT. En effet

cette sortie se fait dans le TURBO-FORTH hôte lequel était en cours d'interprétation de META-BAT. De là, un deuxième BYE ou F8 permet enfin de revenir au DOS et de lancer le nouveau système métacompilé.

Les options du métacompilateur seront modifiées avant le chargement du source dans le cas d'une autre utilisation que la méta-reproduction de TURBO-FORTH.

L'option WIDTH, variable de META, précise la longueur maximale des champs noms de la cible. Une valeur à zéro (META WIDTH OFF) signifie une métacompilation sans en-têtes.

L'option CHECKING ON place le métacompilateur en mode contrôlé asservi au mots marqués de FORTH. CHECKING OFF est l'option par défaut, sans contrôle.

L'option SAVING OFF permet de ne pas sauvegarder la cible en fin de métacompilation sur le mot END-META. Ceci est requis si l'on compte relancer META pour métacompiler une suite au KERNEL. L'option SAVING ON est celle par défaut.

L'option TARGET$ est une variable-chaîne contenant le nom du fichier de sauvegarde si SAVING est ON. On l'affecte par une commande du type " PROG.COM" TARGET$ $! . La variable TARGET$ est par défaut initialisée avec le nom TEST.COM utilisé par la procédure complète META-BAT.

Le rappel des options du métacompilateur s'obtient avec le mot OPTIONS de META. La fin d'une session de méta-compilation s'effectue avec END-META qui affiche un compte-rendu et sauve éventuellement le code métacompilé. On vérifiera surtout qu'il ne reste pas des références FORWARD non résolues. Le mot .SYMBOLS peut en outre être utile pour lister la liste des mots métacompilés et validés.

11. EXEMPLES

La meilleure façon de se familiariser avec les procédures de métacompilation consiste à lire et comprendre les sources. Voyez tout d'abord comment recréer tout TURBO-FORTH avec la commande

INCLUDE META-BAT

Lisez META-BAT.FTHO, examinez META.FTH, parcourrez KERNEL.TXT et NOYAU.TXT.

Voyez ensuite comment créer un petit COMPACT QUESTION.COM en lançant la commande

INCLUDE QST-BAT

Ce petit compact sera bien utile dans vos fichier batch: il permet de poser une question à laquelle on répondra par O ou N. Il reconnait aussi le Y de YES et possède une réponse de normand ??? qui permet de choisir une réponse par défaut ou de reposer la question. Il fournit le paramètre 0 1 ou 2 pour la commande ERRORLEVEL.

L'application est écrite et commentée dans QUESTION.FTH. C'est volontairement qu'est utilisée une structure CASE...ENDCASE pour montrer que PHENIX est intégral: il sait métacompiler cette structure de contrôle.

NOTE DE LA REDACTION:

Le programme de méta-compilation et les nouveaux fichiers source sont diffusés sur la disquette contenant le module M5 de TURBO-Forth, dont voici le contenu:

META	FTH	Nouveau méta-générateur
PHENIX	DOC	L'article ci-dessus
META-BAT	FTH	Le programme de reconstruction de TURBO
KERNEL	TXT	1ère partie du source de TURBO
NOYAU	TXT	2ème partie du source de TURBO
QST-BAT	FTH	Petit programme illustrant COMPACT
QUESTION	FTH	Programme compilé par QST-BAT.FTH
COMPACT1	FTH	Scrutateur récursif de primitives.

Pour toute question, 3615 SAM*JEDI dans les BALs FORTH7 ou SECRETAIRE.

Le programme COMPACT étant d'intérêt général et applicable à d'autres systèmes FORTH 83-Standard, en dehors de TURBO-Forth, il est diffusé ci-après. ■

FORTH

COMPACT-1:
Scrutateur récursif de primitives pour métacompilation de modules COMPACTS

par Michel ZUPAN

Diffusion: module M5 de TURBO-Forth
téléchargement 3615 SAM*JEDI

Listing: COMPACT1.FTH

```
ONLY FORTH DEFINITIONS ALSO HIDDEN ALSO DECIMAL

\ Parcours de l'ensemble du dictionnaire
\ --------------------------------------
DEFER EFFECT ( cfa -- )
\ mot d'action sur chaque mot ( depuis le cfa )

:  APPLY     ( cfa -- )
\ action appliquée à tous les mots des vocabulaires
   IS EFFECT
   VOC-LINK @
     BEGIN ?DUP WHILE
           DUP #THREADS 2* -
           HERE #THREADS 2* CMOVE
           BEGIN   HERE #THREADS  LARGEST   DUP   WHILE
                   DUP LINK> EFFECT
                   @ SWAP !
           REPEAT   2DROP  @
     REPEAT  ;
 \ APPLY permet toutes sortes d'opérations sur
 \       l'ensemble du dictionnaire.
 \ Ainsi pour afficher tous les mots, suffit-il de définir:
 \     : .WORD ?CR >NAME .ID ;   puis d'exécuter
 \     ; .WORD APPLY

 \ Décompilateur récursif et marquage des
 \ primitives dans les VFA
 \ -----------------------------------------

 : +WORD       (S IP -- IP' )   2+ ;
 : +INLINE     (S IP -- IP' )   2+ 2+ ;
 : +STRING     (S IP -- IP' )   2+ COUNT + ;
 : +DOES?      (S IP -- IP' )   DOES? 0= IF DROP 0 THEN ;
 : +FINISH     (S IP -- IP' )   DROP 0 ;

 \ Rappel : Classification de chaque mot dans une
 \ définition
 \ 14 ASSOCIATIVE: EXECUTION-CLASS
 \      (  0 ) ' (LIT) ,            (  1 ) ' ?BRANCH ,
 \      (  2 ) ' BRANCH ,
 \      (  3 ) ' (LOOP) ,           (  4 ) ' (+LOOP) ,
 \      (  5 ) ' (DO) ,
 \      (  6 ) ' COMPILE ,          (  7 ) ' (.") ,
 \      (  8 ) ' (ABORT") ,
 \      (  9 ) ' (;CODE) ,          ( 10 ) ' UNNEST ,
 \      ( 11 ) ' (") ,
 \      ( 12 ) ' (?DO) ,            ( 13 ) ' (;USES) ,

 15 CASE: +EXECUTION-CLASS
      (  0 )  +INLINE               (  1 )  +INLINE
      (  2 ) +INLINE                (  3 )  +INLINE
      (  4 )  +INLINE               (  6 )  +INLINE
      (  6 )  +WORD                 (  7 )  +STRING
      (  8 ) +STRING                (  9 )  +DOES?
      ( 10 )  +FINISH               ( 11 ) +STRING
      ( 12 )  +INLINE               ( 13 )  +FINISH
      ( 14 ) +WORD    ;

 : NEXTWORD ( IP -- IP' )
     DUP @ EXECUTION-CLASS +EXECUTION-CLASS ;

 : SPOT    ( cfa-- ) >VIEW ON  ;  \ marque un mot
 : UNSPOT  ( cfa-- ) >VIEW OFF ;
   \ efface la marque d'un mot
```

```
: .SPOT ( cfa-- )                 \ affiche un mot si marqué
  DUP >VIEW @
  IF
    DUP @ ['] KEY @ OVER = SWAP ['] EMIT @ = OR
    IF BOLD THEN \ vectorisé
    DUP ['] START > IF UNDERL THEN  \ hors KERNEL
     ?CR >NAME .ID
  ELSE DROP
  THEN ATTOFF ;

: LOOKUP ( cfa-- )
  \ marque tous les mots utilisés par un mot
  DUP SPOT
  DUP @ ['] QUIT @ = IF >BODY
     BEGIN DUP @ RECURSE NEXTWORD DUP 0= UNTIL
     THEN DROP ;

: FOREWORDS ( <mot> -- )
  \ affiche toutes les primitives d'un mot
  ['] UNSPOT APPLY
  \ remet toutes les marques à zéro
  ' LOOKUP CR
  \ marque les primitives du mot qui suit
  ['] .SPOT  APPLY          \ affiche les mots marqués
;

\ FOREWORDS <MOT>  affiche  toutes  les  primitives
\ utilisées par  <MOT>
\ Les mots vectorisés sont en gras, les mots qui
\  ne  font  pas  partie  du KERNEL de TURBO-FORTH
\  sont soulignés. L'édition d'un noyau ne comportant
\  que ces mots, puis métacompilé en  WIDTH  OFF,
\  produit  un  COMPACT  de l'application <MOT> sans
\  en-têtes ni librairie.

EOF
```

Un Programme d'Application Compact ou COMPACT est un module Forth compilé sans en-têtes ni librairie. Un tel programme s'utilise dans Forth mais surtout en dehors de Forth comme un quelconque programme compilé en code. L'avantage du compactage est évident: chargement rapide, gain de place dans la mesure où seules les primitives de l'application sont compilées sans dictionnaire, protection considérablement accrue du fait du caractère "fermé" de l'application sans possibilités pratiques de décompilation ou de désassemblage.

Je vous propose ici une méthode de compilation de COMPACTS assistée par l'utilisation d'un "scrutateur récursif de primitives". L'écriture du source d'un COMPACT reste cependant en grande partie manuelle. En fin d'article j'aborderai l'étude de la surgénération automatique de COMPACTS.

Rien de tel qu'un exemple pour illustrer notre propos: j'ai choisi une application volontairement simpliste qui ne tiendra qu'en un mot Forth. Supposons que nous désirions fréquemment connaître la liste des commandes exécutables dans le répertoire courant c'est-à-dire le directory des fichiers .COM ou .EXE ou encore .BAT qui sont les seules extensions autorisées dans cet ordre de recherche par MS-DOS pour des commandes directes. Le mot DIRCOM réalise facilement cet objectif. En voici une version simplifiée qui affiche les noms des fichiers avec extensions et tailles sans dates ni heures de créations:

```
: DIRCOM  ( --- ) \ affiche directory des fichiers .COM
.EXE .BAT
 128 SET-DMA  ." Fichiers exécutables:"
 " *.BAT" " *.EXE" " *.COM"
 3 0 DO
      PAD SWAP CMOVE 0 PAD 5 + C! \ chaîne ascii0
        PAD 16 (SEARCH0)      \ cherche 1er fichier
      IF                      \ si trouvé:
         BEGIN CR .NAME       \ affiche nom, ext, taille
               (SEARCH)       \ cherche un autre
         NOT UNTIL
      THEN
    LOOP ;
```

La solution classique pour définir un module d'application Forth consiste maintenant à sauver le programme tel quel. L'exécution en sera immédiate avec sortie dès celle-ci achevée:

```
: APPLICATION DIRCOM BYE ;
' APPLICATION IS BOOT
SAVE-SYSTEM DIRCOM.COM
```

Nous disposons ainsi d'un module exécutable DIRCOM autonome qui peut être lancé depuis le prompt du DOS ou d'un fichier batch, depuis un programme BASIC (SHELL "DIRCOM"), un programme dBASE (RUN DIRCOM), PASCAL, C, TURBO-FORTH (PROGRAM DIRCOM.COM ou " DIRCOM" SHELL), ou depuis tout autre logiciel possédant un intégrateur.

DIRCOM.COM est une commande d'environ 26K. Ce n'est pas énorme mais vous comprenez aisément tout l'intérêt qu'il y aurait à disposer du même programme compilé sans en-têtes avec les seules primitives de notre mot Forth d'application. Songez à la place occupée dans ces 26K par l'interpréteur Forth, le compilateur, l'assembleur, les utilitaires et des centaines de mots inutiles pour notre programme.

Pour compacter DIRCOM, il "suffit" de métacompiler sans en-têtes un noyau Forth ne comportant que ce qui est nécessaire au mot DIRCOM. La métacompilation sans en-têtes est autorisée dans le métacompilateur: avec comme directive de métacompilation une largeur de noms WIDTH à zéro, les VFA, LFA, ou NFA ne seront pas compilés dans la cible. Pour savoir quelles sont les primitives de DIRCOM à compiler, nous utiliserons un scrutateur récursif de primitives qui va "marquer" tous les mots utilisés par DIRCOM dans un processus de décompilation récursive jusqu'aux mots de plus bas niveau.

APPLY permet d'appliquer sur tous les mots de tous les vocabulaires une procédure quelconque. Les applications dépassent de loin notre propos actuel: je vous laisse le loisir d'y méditer.

Définissons maintenant notre scrutateur de primitives. Nous récupérons pour ce faire une partie du décompilateur dans le vocabulaire HIDDEN. Pour marquer (SPOT) un mot, nous utiliserons pour chaque mot ce champ d'indexation qu'est le view-field. J'insiste sur l'importance qu'il y a à considérer les view-fields (vfa) comme des pointeurs multi-usage. Bien entendu, nous perdrons temporairement l'utilitaire d'autodocumentation mais quelques LEARN pourront par la suite le rétablir.

Le coeur du système est LOOKUP qui décompile et marque toutes les primitives d'un mot de façon récursive. Le mot utilisateur est FOREWORDS qui liste le travail de LOOKUP.

Si nous exécutons FOREWORDS DIRCOM nous obtenons la liste suivante des primitives de notre application:

<u>DIRCOM</u> .NAME DMA SET-DMA (.") (") SCAN D.R (D.) #S # SIGN #> <# HOLD BDOS (SEARCH) (SEARCH0) SPACES REPLICATE TYPE CR PAD HERE COUNT BL EMIT HLD BASE MU/MOD DABS 2DUP 2DROP 2@ MAX < 0< 0= UM/MOD 1-! 1- 1+ 3 0 - + NOT OR AND R@ >R R> ?DUP -ROT ROT TUCK OVER SWAP DUP DROP CMOVE C! C@ ! @ (?DO) (DO) (LOOP) ?BRANCH BRANCH (LIT) UNNEST

DIRCOM apparaît souligné: c'est le seul dont la définition n'est pas dans le KERNEL de TURBO-FORTH. CR et EMIT sont en gras: ces mots vectorisés devront recevoir un vecteur approprié à l'application.

Ici s'achève l'aide à la métacompilation du COMPACT. Muni d'un bon éditeur, reprenons KERNEL.TXT pour garder ces seules primitives, ajoutons la définition de DIRCOM et définissons simplement COLD comme DIRCOM BYE.

Nous éditons ainsi un fichier DIRCOM.TXT qu'il ne reste plus qu'à métacompiler sans en-têtes:

```
INCLUDE META
META WIDTH OFF
INCLUDE DIRCOM.TXT
```

Pour vous fixer un ordre de grandeur, le fichier DIRCOM.TXT que j'ai édité sans chercher à optimiser fait

11K et le COMPACT produit ainsi est un fichier DIRCOM.COM qui ne fait plus que 1,2K soit un rapport de compactage de un à vingt. Le gain est évidemment très appréciable...

Notre prochaine étape sera, une fois l'application Forth marquée par le décompilateur récursif, d'automatiser la surcompilation du COMPACT. La solution passe par un métacompilateur qui ne compilera dans la cible que les mots marqués dans l'hôte.

INFORMATION

SEMINAIRES ET CONGRES INFORMATIQUES EN EUROPE pour la saison 88-89

12-16 Septembre 1988
EUROGRAPHICS'88 - NICE (F)
Organisateurs: INRIA,EUROGRAPHICS. Renseignements et inscriptions: INRIA Relations exterieures. Tel. 39 63 55 01

12-16 Septembre 1988
Convention IA'88 / 1ere conference et exposition europeenne sur les techniques et les applications de l'IA - PARIS (F)
Organisateurs: BIRP. Renseignements et inscriptions: BIRP, Ed. Hermes

12-16 Septembre 1988
Ecole d'ete de conception optimale de structures assistees par ordinateur - SOPHIA-ANTIPOLIS (F)
Organisateurs: CCE. Renseignements et inscriptions: Fac. des Sciences, IMSP. Dept. de Math. Parc Valrose 06034 Nice. Tel. 93 52 98 98

15-16 Septembre 1988
3rd Conference on modelling techniques and tools for performance evaluation - PALMA DE MAJORQUE (SP)

15-17 Septembre 1988
ECOOP'88 / European conference on object-oriented programming - OSLO (N)

19-22 Septembre 1988
CRIS'88 / Conference on computerized assistance during the information systems life cycle - EGHAM (GB)

20-23 Septembre 1988
SICOB-Congres INFODIAL-VIDEOTEXT / L'Industrie de l'information et du videotex en Europe a l'horizon 1992- PARIS (F)
Organisateurs: SICOB. Renseignements et inscriptions: SICOB

21-23 Septembre 1988
ESSCIRC'88 / 14th European solid-state circuits conference - MANCHESTER (GB)

24 Septembre - 01 Octobre 1988
1st International symposium on electronic art - UTRECHT (NL),AMSTERDAM (NL),ROTTERDAM

26-28 Septembre 1988
Journees "Systemes experts et gestion d'entreprises" / SEGE'88 - VERSAILLES (F)
Organisateurs: EC2. Renseignements et inscriptions: M.-M. Sainflou, 269-287 Rue de la Garenne. 92000 Nanterre. Tel. 47 80 70 00

26-30 Septembre 1988
2e Journees/ Pratique des methodes et outils logiciels d'aide à la conception de systemes d'information - NANTES (F)
Organisateurs: LIANA (Univ. de Nantes. IUT Lab. d'informatique). Renseignements et inscriptions: H. Habrias, IUT, LIANA, 3 rue Mal Joffre. 44041 Nantes. tel. 40 30 60 90

03-07 Octobre 1988
6e Colloque international de fiabilite et de maintenabilite - STRASBOURG (F)
Organisateurs: CNES. Renseignements et inscriptions: ADERA. BP 48. 33166 St Medard en Jalles Cedex. Tel. 56 05 84 24

03-07 Octobre 1988
EUUG Autumn'88 conference / new directions for UNIX- LISBONNE (P)

04-06 Octobre 1988
Colloque ergonomie et intelligence artificielle - BIARRITZ (F)
Organisateurs: AFCET. Renseignements et inscriptions: AFCET

17-19 Octobre 1988
4th SAS World conference-exposition on structural analysis CAD/CAM and computer graphics / FEMCAD'88 - PARIS (F)
Organisateurs: IITT. Renseignements et inscriptions: IITT (Institute for industrial technology transfer)

17-20 Octobre 1988
EUSIDIC annual conference / Information-now - HEIDELBERG (D)

17-21 Octobre 1988
3rd International symposium on knowledge engineering- MADRID (SP)

18-20 Octobre 1988
7e Conference annuelle europeenne / prise de decision et controle manuel - PARIS (F)
Organisateurs: EDF. Renseignements et inscriptions: EDF-Clamart (Mr Lhory)

24-28 Octobre 1988
ISATA / 19th International symposium on allied technology and automation - MONTE CARLO (MC)

15-17 Novembre 1988
Journees internationales / Les reseaux neuro-mimetiques et leurs applications / NEURO-NIMES'88 - NIMES (F)
Organisateurs: EC2. Renseignements et inscriptions: EC2, 269 Rue de la Garenne 92000 Nanterre. tel. 47 80 70 00

16-18 Novembre 1988
7th International conference on entity-relationship approach / ER'88 - ROME (I)

05-09 Decembre 1988
Journees internationales / le genie logiciel et ses applications - TOULOUSE (F)
Organisateurs: EC2. Renseignements et inscriptions: EC2, 269 Rue de la Garenne 92000 Nanterre. tel. 47 80 70 00

06-08 Decembre 1988
12th International Online information meeting / ONLINE'88 - LONDRES (GB)

07-09 Decembre 1988
Journees nationales "intelligence artificielle" / perspectives pour l'education et la formation - LYON (F)
Organisateurs: INRP (Institut national de recherche pedagogique). Renseignements et inscriptions: B. Dumont, INRP, 91 rue Gabriel Peri. 92120 Montrouge

13-15 Mars 1989
2e Colloques national / l'automatique pour l'aeronautique et l'espace - PARIS (F)
Organisateurs: SMAI (Societe des mathematiques appliquees et industrielles). Renseignements et inscriptions: CNRS-LSS/ESE. Plateau du Moulon. 91190 Gif sur Yvette. Tel. 69 41 80 40

11-13 Avril 1989
4th International conference on - developments in power system protection - EDINBURGH (GB)

11-15 Septembre 1989
5th International symposium on numerical methods in engineering - LAUSANNE (CH)
Organisateurs: EPFL,GAMNI. Renseignements et inscriptions: S. Nicli. GASOV-EPFL. CH 1015 Ecublens (CH). Tel. (41)21-47-22-11

```
*               ┌─ dBASE III+ ──────────────────────────────┐
*               │                                           │
*               │              MENU DEROULANT               │
*               │                                           │
*               └───────────────────────────────────────────┘
*                             par Marc PETREMANN
*
* diffusion 3615 SAM*JEDI
* adaptable dBASE III sans difficulté
*
DO WHILE .T.
SET COLOR TO W+/N
@ 6,9 TO 16,74 DOUBLE
@ 6,20 SAY [ E D I T I O N   D E S   R A P P O R T S ]
SET COLOR TO W/N
ch0 = [ FIN DES EDITIONS                                               ]
ch1 = [ RECHERCHE ET EDITION DES ERREURS DE SAISIE                     ]
ch2 = [ EDITION DES RECAPITULATIFS PAR TYPE DE CARBURANT               ]
ch3 = [ EDITION DES SUIVIS MENSUELS PAR TYPE DE CARBURANT              ]
ch4 = [ EDITION DU RECAPITULATIF MENSUEL ACHATS CARBURANT ET DISTANCE  ]
ch5 = [ EDITION D'UN BORDEREAU DE SAISIE DES CONSOMMATIONS             ]
ch6 = [ STATISTIQUES DE LA CONSOMMATION D'UN VEHICULE                  ]
ch7 = [ STATISTIQUES SUR L'ENSEMBLE DU PARC VEHICULES                  ]
ch8 = [ EDITION CONDUCTEUR, CONSOMMATION ET REPARATIONS MENSUELLES     ]
SET INTENSITY ON
@ 07,10 SAY ch0
@ 08,10 SAY ch1
@ 09,10 SAY ch2
@ 10,10 SAY ch3
@ 11,10 SAY ch4
@ 12,10 SAY ch5
@ 13,10 SAY ch6
@ 14,10 SAY ch7
@ 15,10 SAY ch8
choixa = 0
xchoixa = STR(choixa,1)
tou = 0
@ 7,10 GET ch0
DO while .T.
   tou = INKEY()
   DO WHILE tou=0
      tou = INKEY()
   ENDDO
   DO CASE
      CASE (tou=5) .OR. (tou=24)
         @ 7+choixa,10 SAY CH&xchoixa
         STORE IIF(tou=5,choixa-1,choixa+1) TO choixa
            IF choixa=-1
               choixa=8
            ENDIF
         choixa = MOD(choixa,9)
         xchoixa = STR(choixa,1)
         @ 7+choixa,10 GET CH&xchoixa
         CLEAR GETS
         LOOP
         * validation par appui sur CR
      CASE tou=13
         EXIT
   ENDCASE
ENDDO

* Le présent programme illustre une procédure de gestion de menu déroulant
* utilisable sous dBASE III+. Le paramètre de sortie est choixa qui sera
* ensuite traité dans une structure de type CASE...ENDCASE.
```

CONTENU DU FORUM 3615 SAM*JEDI

PILVERDIER Du 11.07.88 A 15h05
DANS F-PACK :
LE MOT ?10PWR NE RENVOIT PAS LA VALEUR A LAQUELLE ON POURRAIT S'ATTENDRE, EST-CE NORMAL VU SON UTILISATION?

SECRETAIRE Du 12.07.88 A 14h06
REPONSE A PILVERDIER, concerne ?10PWR Ce mot n'est utilise que dans OPREP lequel est utilise dans (E.) et (F.) Les mots (E.) et (F.) sont les primitives de E. et F., lesquels ont l'air de fonctionner correctement:

```
 F' 10 F.    affiche  10
```

S'il y a des problemes avec ces mots, preciser dans quelle situation ils se produisent. Tenir compte des valeurs limite des grandeurs utilisees apres F'.

SECRETAIRE Du 21.07.88 A 13h27
BLACKOUT, PROGRAMME ENVOYE PAR M. ZUPAN: Eteint l'ecran sans perte de l'affichage en cours. Economise votre ecran lors des pauses cafe ...
Ne fonctionne pas obligatoirement sur tous les systemes PC
LISTING:

```
hex
: BLACKOUT ( -- ) \ inhibe la video
  40 65 LC@ F7 AND 40 63 L@ 4 + PC! ;
: LIGHTUP  ( -- ) \ retablit la video
  40 65 LC@  8 OR  40 63 L@ 4 + PC! ;
\ Utilisation:
\  bascule video on/off avec ctrl-V
VARIABLE LIGHTING TRUE LIGHTING !
: V-IN LIGHTING @ DUP
  IF BLACKOUT
  ELSE LIGHTUP THEN NOT LIGHTING ! ;
' V-IN CC @ CONTROL V 2* + !
decimal
EOF
```

C'est tout!.

FORTH7 Du 21.07.88 A 16h20
Je recidive: apres quelques deboires (apologies to sysop) je vous propose en un seul mot Turbo-Forth de balancer un fichier texte quelconque dans CE forum: syntaxe: FORUM (fichier.txt).
Les lignes sont tronquees a 39 car. et suivies de SUITE. Les accents et autres non ascii sont remplac*es par *. Un point et ENVOI sur la derniere ligne.
La partie en code configure votre RS232 (COM1) directement en mode minitel sans avoir a utiliser MODE. J'ai mis plein de tempos a case de l'oncle SAM...

```
\ forum SAM*JEDI seulement
only forth also definitions decimal
: FORUM ( (fichier) -- )
asm[ 154 # ax mov   0 # dx mov
     20 int   next   ]forth
handle @ open handle !
begin getline buffer count 39 min
0 ?do count dup 32 127 between not
if drop ascii * then 4 (out) 150 ms
loop drop 19 4 (out) 72 4 (out)
7000 ms eof? @ until
ascii . 4 (out) 19 4 (out) 65 4 (out)
close handle ! ;
```

Quelques precisions encore sur FORUM:
1)Les lignes sont impitoyablement limitees aux 39 premiers caracteres: concoctez votre prose en consequence!
2)Les temporisations sont larges: 150 millisecondes entre chaque car. pour absorber les 75 bauds et 7 secondes en fin de ligne pour laisser SAM faire son menage. Notre SYSOP nous dira si on peut faire plus vite. Reglez toutefois FUDGE pour que vos millisecondes ne durent pas un quart d'heure: 30 FUDGE ! pour un systeme a 4,7 MHz est une valeur approchee satisfaisante.

F32 Du 25.07.88 A 10h54
DE:F32 LE 25/7/88
A :SECRETAIRE
Y A-T IL DES REACTIONS SUR F32 ?
J'AI LE TEXTE DU BREVET EUROPEEN POUR LE NC 4016 EXTREMEMENT DETAILLE POUR LES INTERESSES ,JE PENSE A G.DUMUR ME LE DEMANDER POUR COURRIER:80 PAGES! DISCUSSION EN COURS AVEC NOVIX POUR LES PROBLEMES DE PROTECTION DES DROITS... UNE CARTE NO4000 CIRCULE-T-ELLE DANS JEDI? CE SERAIT LE MEILLEUR EMULATEUR SUIS TOUJOURS A LA RECHERCHE D'UNE STE FRANCAISE POUR UTILISER LE MICRO. C'EST PLUTOT DESERTIQUE! UN MEMBRE CONNAIT-IL UNE BOITE UN PEU DYNAMIQUE? +

GRIMAULT Du 26.07.88 A 13h43
QUELQU'UN PEUT-IL M'ECLAIRCIR SUR LES DIFFERENCES ENTRE LES MICROS NC4016 (NOVIX) ET RTX2000 (HARRIS) ?
PLUS PARTICULIEREMENT, QUEL FORTH EST UTILISE PAR LE RTX2000?

SECRETAIRE Du 28.07.88 A 14h27
AUTRE VERSION >FORUM DE FORTH7
Dans le fichier RSINIT, je propose une application permettant de transmettre des caracteres et codes de controle TELETEL vers le MINITEL. Dans la partie documentation, un exemple de connexion SAM*JEDI est donne. Voici une version modifiee de EMET, que je nomme TRANSMET et qui envoie une chaine ASCII vers le MINITEL:

```
: TRANSMET ( adr long ---)
   [ ' EMIT >BODY @ UP @ + ] LITERAL @  >R
   ['] (RSEMIT) IS EMIT  TYPE  R> IS EMIT ;
```

Ce mot est utilise en le faisant preceder de l'adresse et de la longueur d'une chaine de caracteres; exemple:

```
  80 STRING A$
  " TEST DE TRANSMISSION " A$ $!
  A$ TRANSMET
```

envoie sur le minitel le contenu de A$.
Maintenant, pour transmettre tout un fichier ASCII dans un message destine a FORUM, utiliser le mot >FORUM dont la definition est la suivante:

```
\ forum SAM*JEDI seulement
VARIABLE #LIGNES
: >FORUM ( --- (fichier) )
MINITEL  HANDLE @ OPEN HANDLE !  0 #LIGNES !
  BEGIN  GETLINE BUFFER COUNT 39 MIN
     TRANSMET SUITE 7 ATTEND  1 #LIGNES +!  STOP?
     IF  SOMMAIRE CLOSE HANDLE ! EXIT  THEN
     #LIGNES @ 19 =
     IF  ENVOI 12 ATTEND  0 #LIGNES !  THEN
  EOF? @ UNTIL
  ASCII . >RS  ENVOI CLOSE HANDLE ! ;
```

Ce mot a l'avantage de transmettre n'importe quel fichier ASCII en pages de 19 lignes. Chaque ligne est suivie de SUITE, chaque page est coupee par ENVOI en fin de fichier, la sequence . et ENVOI termine le message. Les accentues sont automatiquement traduits en leur equivalents non-accentues.

SECRETAIRE Du 29.07.88 A 13h13
AFFECTATION D'UNE CHAINE ALPHANUMERIQUE DEPUIS LA LIAISON SERIE: Lors d'un acces MINITEL-PC, il peut etre interessant de recevoir une serie de caracteres dans le sens MINITEL vers PC.
Exemple: lors d'une tentative d'acces sur le 3614, le PAV repond en general par une premiere sequence du type LIB XX. Je propose la definition du mot RSINPUT qui attend un nombre de caracteres limite en provenance de la liaison MINITEL-PC:

```
VARIABLE SORTIE
: RSINPUT ( adr long ---)
   SORTIE OFF    DROP DUP 2- C@ OVER  SWAP OVER +  SWAP
   DO BEGIN  OR [GET] 1 = (KEY?) OR UNTIL
    RX [GET]  I C!  \ stockage caractere
      (KEY?)  IF I OVER - SWAP 1- C! SORTIE ON LEAVE THEN
   LOOP  SORTIE @ IF EXIT ELSE DUP 2- C@ SWAP 1- C! THEN ;
```

Ce mot doit etre precede d'une variable alpha-numerique:

```
12 VARIABLE A$
A$ RSINPUT
```

Pour exemple, vous avez allume votre MINITEL, compose le 3614, puis envoye une sequence de neuf chiffres du style 135000171 par la procedure suivante:

```
 EMET" 135000171" ENVOI
 A$ RSINPUT
 A$ TYPE
```

affichera les douze premiers caracteres transmis par

TRANSPAC apres ENVOI. Ces caracteres sont contenus dans A$. Si la sequence contient LIB XX XXX, vous pouvez faire un programme envoyant le cas echeant ANNULATION ou CORRECTION ou CXF.

FORTH7 Du 29.07.88 A 18h08
L'auteur de ET sort une nouvelle version de cet editeur de textes en francais pour programmeurs distribue en shareware: plein ecran ultra rapide, 8 fichiers ASCII simultanes, blocs, recherches substitutions, tabulations, partition ecran, multivideo, macros editables, reconfigurations clavier, aide, integrateur dos et j'en passe. Cette version permet le positionnement ligne colonne au lancement ce que seul l'éditeur de M.FAIVRE pour Turbo-Forth nous offrait. ET 1.5 ne fait que 25K! En attendant que les forthiens se décident à écrire un EDIT-TOOLBOX en Forth pour fichiers ASCII afin que chacun ait l'editeur de ses reves.

GUILLAUMAUD PH. Du 30.07.88 A 11h06
BONJOUR . J'AI ECRIS UNE VERSION POUR PC DU LANGAGE FORTH , ET JE LA TIENS A LA DISPOSITION DES AMATEURS DE FORTH .
POUR TOUS RENSEIGNEMENTS :
MR. GUILLAUMAUD PHILIPPE 4 AVENUE JEAN MOULIN 93140 BONDY
TELEPHONE : 48 48 79 45. JE DISPOSE AUSSI D'UNE B.A.L. SUR 3615 SAM*JEDI, CODE PHIL. QUE LE FORTH SOIT AVEC VOUS...

SECRETAIRE Du 31.07.88 A 01h20
MODIFICATION DU VECTEUR D'INTERRUPTION DE CLAVIER SUR PC:
Je suis tombe dernierement sur un os, a savoir que sur le systeme de traitement de texte OLIVETTI ETV 260 (compatible PC...), le clavier n'est pas gere de maniere conventionnelle. L'activation de KEYBFR.COM avant execution d'un programme autre que le traitement de texte integre, detruit les affectations des touches "Traitement texte". Pour les retablir, il a fallu restaurer le vecteur d'interruption clavier a son etat initial.
Moi, vous me connaissez, passionne par TURBO-Forth, je vous ai resolu ce dilemne a ma maniere. Donc, je lance TURBO et regarde le contenu de l'adresse 0000:0024 contenant le segment et l'offset de la routine de decodage du clavier. Ce contenu est lu avant execution de KEYBFR par:

```
  HEX 0 24 L@ U. CONSTANT N1
      0 26 L@ U. CONSTANT N2
```

Puis on lance KEYBFR par les commandes suivantes:

```
  PROGRAM C:chemin\KEYBFR.COM
  ( preciser chemin le cas echeant)
```

En controlant le contenu de 0 24 L@ et 0 26 L@ (en hexa), on constatera que ces valeurs ont effectivement ete modifiees par rapport au contenu de N1 et N2.
Maintenant, creons on programme qui sera charge de retablir le vecteur d'interruption d'origine. Oh, il ne suffit pas de faire N1 0 24 L!, car il y aura plantage garanti. Il faut masquer les interruptions par le code assembleur cli. Voici le detail de la procedure a compiler pour arriver a executer cette delicate manoeuvre:

```
\ Programme RESTAURE.FTH
HEX 0 24 L@ CONSTANT N1
    0 26 L@ CONSTANT N2
: RESTAURE ( ---)
  ASM[ cli next ]FORTH  N1 0 24 L!  N2 0 26 L!
  ASM[ sti next ]FORTH  BYE ;
' RESTAURE IS BOOT
CR .( Taper SAVE-SYSTEM RESTAURE) CR
```

Sauvez ceci dans un fichier nomme RESTAURE, puis sous TURBO, tapez INCLUDE RESTAURE SAVE-SYSTEM RESTAURE BYE
Maintenant, vous pouvez sur votre ETV 260 lancer KEYBFR et restaurer le clavier du traitement de texte integre en tapant RESTAURE. Merci TURBO!

SECRETAIRE Du 31.07.88 A 01h58
CONCERNANT >RS DANS FICHIER RSINIT.FTH:
Il semble que ce mot ne se comporte pas de maniere identique sur toutes les liaisons MINITEL PC. Il y a bourrage lors d'une transmission de chaine trop longue. Voici le remede:

```
72 CONSTANT TEMPO
: >RS ( c ---)
   TX [PUT] \ envoi car vers reg de transmission
   BEGIN  THRE [GET]
   \ attente reg de transmission vide
   UNTIL TEMPO MS ;
```

On ajustera le temps entre deux caracteres par:
n IS TEMPO ou n est une valeur delivree par TEMPO lors de son execution. Avec la valeur 72, on peut abaisser la valeur d'attente d'envoi entre deux lignes dans la definition de >FORUM donnee dans un precedent message.

FORTH7 Du 31.07.88 A 14h45
En reponse a la question de F32, Des piles Forth de profondeur 256 sont bien suffisantes: certains forth n'en n'ont pas la moitie. Empiler 256 parametres est souvent une faute de programmation et la pile de retour est rarement sollicitee au dela d'une dizaine de cellules sauf dans le cas d'une recursion vraie (tours de Hanoi a 256 plateaux !). Si la question porte sur le fait de savoir si un chip avec 256 mots-instructions Forth suffit, je signale que Turbo-Forth comporte 168 mots en code dont des constantes codes et pas mal d'interfaces MS-DOS pour des raisons de rapidite 8086. La machine virtuelle Forth n'a guere besoin que d'une cinquantaine de primitives!

FORTH7 Du 31.07.88 A 14h50
Le dernier numero de Forth Dimensions rapporte les travaux du comite ANS de standardisation du Forth: le Forth-83 est adopte comme base mais la taille du 'mot' n'est plus machine-dependante ce qui ouvre la porte a des systemes sur mots 32-bits. NIP et TUCK sont homologues. La division sera plancher ou non avec possibilite de passer d'un mode a l'autre pour garder la compatibilite entre diverses sources. Un nouveau concept d'interpretation de chaines arbitraires est introduit sous forme du mot EVAL dont je n'ai pu trouver la definition: ne s'agirait-t-il pas tout simplement de $EXECUTE ???

SECRETAIRE Du 03.08.88 A 00h57
ESSAI FORTH DE GUILLAUMAUD
Bon, ben j'ai teste ce nouveau FORTH et suis assez etonne. Pour le moins bien, disons qu'il est nettement incomplet. Il manque toute la panoplie d'aide au developpement dont dispose TURBO.
Pour le mieux: il decompile en affichant les structures sous forme IF..THEN, BEGIN..UNTIL et avec les indentations, svp. Sachant que ce decompillateur est INTEGRALEMENT ecrit en code machine, je tire mon chapeau. Ce nouveau FORTH, nomme Le Forth par ses auteurs (fanatiques egalement de LISP... ils vont s'entendre avec Mr JACCOMARD), compile des fichiers ASCII mais sans imbrications. Bref, un produit curieux qui fait penser au JUPITER ACE.
Si vous le voulez, demandez en un exemplaire a GUILLAUMAUD PH. (dans sa BAL), ou envoyez-nous 10 timbres a 3,70 au siege de l'association.

GUILLAUMAUD PH. Du 04.08.88 A 08h45
LES CRITIQUES SONT TOUJOURS BONNES QUAND ELLES PERMETTENT DE FAIRE EVOLUER UN PRODUIT .
LA VERSION 2 DE LE FORTH EST EN COURS DE DEVELOPPEMENT , ET SERA CONFORME AU STANDARD 83, CE QUI EST LA MOINDRE DES CHOSES... MERCI A MARC POUR CE BANC D'ESSAI. QUE LE FORTH SOIT AVEC VOUS

SECRETAIRE Du 04.08.88 A 19h39
FONCTIONS EVAL CONTRE $EXECUTE
Dans le magazine Dr.Dobb's Journal d'octobre 87, Lori CHAVEZ, un americain bien au courant de ce qui se fait de mieux dans le domaine du standard FORTH definit la fonction EVAL:

```
 : EVAL ( adr long ---)
 DUP >R TIB SWAP CMOVE R@ #TIB !
 0 >IN ! 0 BLK ! INTERPRET R> >IN ! ;
```

et peut etre utilisee sous la forme suivante:

```
 : MOI
  " FORGET MOI" EVAL ;
```

Or, cette fonction EVAL a pour moi un air de deja vu. Bon sang, mais c'est bien sur! c'est $EXECUTE de TURBO-Forth! Aurions-nous eu la meme idee en France et Outre-Atlantique? Oui, mais...
Le mot $EXECUTE de TURBO-Forth est re-entrant lui! Voici un petit exemple:

```
  80 STRING B$     80 STRING A$
  " DARK WORDS " B$ $!
  " B$ $EXECUTE BELL " A$ $!
```

puis entrons A$ $EXECUTE. Le contenu de A$ s'execute, c'est a dire execute le contenu de B$. En fin de B$, la

suite de A$ s'execute. La re-entrance de $EXECUTE de TURBO-Forth n'est limitee que par la profondeur de pile.

FORTH7 Du 06.08.88 A 13h34
TF83 Bogue TF83 Bogue TF83 Bogue TF83

Dans (GET) et (PUT) qui lit/ecrit dans un fichier (segment tampon long hndl-- long.lue/ecrite flag) il faut lire a la deuxieme instruction AX ES MOV au lieu de ES AX MOV. Le crash se produit quand on travaille en extra-segment entre deux (GET) ou (PUT) car ES n'est pas toujours identique a DS. Il n'est pas necessaire de re-metacompiler pour corriger cette bogue :

```
hex 8E ' (GET) 4 + C!
    8E ' (PUT) 4 + C!  decimal
( remplace 8C par 8E dans les 2 mots )
avant SAVE-SYSTEM TURBO
```

SECRETAIRE Du 07.08.88 A 12h43
A PAUL ORTAIS; concerne brevet NC4016: 80 pages c'est assez volumineux. Afin de faire progresser le projet F32, serait-il possible d'en faire une syn- these qui sera diffusee dans JEDI?
Je sais, ecrire prends du temps, mais n'est ce pas preferable a reproduire en x exemplaires une doc volumineuse.

FREDB Du 12.08.88 A 20h39
UEBER SQRT IN F83 HANDBUCH:
(BTX --> TELETEL, PRUEFUNG...)
DAS WURZELPROGRAM

```
  : SQRT  DUP 2/ 10 0 DO OVER OVER /
  + 2/ LOOP SWAP DROP ;
```

GEHT NUR FUER ZAHLEN BIS 32767. DIE FOLGENDE MODIFICATION GEHT BIS 65535:

```
  : SQRT  DUP U2/ 10 0
  DO 2DUP 0 SWAP UM/MOD SWAP DROP + U2/
  LOOP SWAP DROP ;
```

SECRETAIRE Du 12.08.88 A 22h03
TIENS, NOUS AVONS UN CONSULTANT OUTRE- RHIN. JE TRADUIS: LE MOT SQRT FIGURANT DANS LE MANUEL F83 NE TRAITE QUE LES NOMBRES DE 0 A 32767. SA DEFINITION MODIFIEE VA JUSQU'A 65535.

SECRETAIRE Du 14.08.88 A 08h41
COMPILATION CONDITIONNELLE
Voici un moyen tres simple de faire de la compilation conditionnelle. On redefinit un mot similaire a \ mais qui reagit en fonction d'un flag booleen:

```
 : ?\ ( fl ---)
   NOT IF [COMPILE] \ THEN ; IMMEDIATE
```

En execution, le mot ?\ compile ou ne compile pas le texte qui le suit:
- flag=0, agit comme \
- flag<>0, agit conne NOOP

Voici un exemple illustrant une utilisation possible de ce mot:

```
  FALSE CONSTANT FRENCH    IMMEDIATE
  FALSE CONSTANT GERMAN    IMMEDIATE
  FALSE CONSTANT ENGLISH   IMMEDIATE
  TRUE IS ENGLISH
  : 1DAY ( ---)
    FRENCH   ?\ ." LUNDI"
    GERMAN   ?\ ." MONTAG"
    ENGLISH  ?\ ." MONDAY"
  ;
```

La decompilation de 1DAY montre que la seule chaine compilee a ete celle validee par ENGLISH. Toute l'astuce consiste ici a exploiter des constantes declarees immediates et dont le contenu est modifiable par une sequence du type
n IS <constante>

F32 Du 19.08.88 A 22h34
FORTH7 SUITE A UNE DE MES QUESTIONS CITE LA TAILLE MINIMALE D'UNE MACHINE VIRTUELLE.
D'OU QUESTION: PEUT-ON DEFINIR DE FACON RIGOUREUSE CETTE MINIMALITE, SI OUI COMMENT, SI C'EST DEJA FAIT, OU ET QUI?

JE SUGGERE QUE LES PERSONNES INTERESSEES PAR F32 DESIGNENT LEUR EMPLACEMENT GEOGRAPHIQUE, QUE L'ON PUISSE SAVOIR QUI OCCUPE LE BARYCENTRE OU DU MOINS LE MEILLEUR ENDROIT POUR UNE RENCONTRE PRELIMINAIRE INDISPENSABLE. CONTACT PHYSIQUE A L'EPOQUE DU MINITEL ! POUAH

JE PARLAIS DU F83 POUR LA MACHINE VIRTUELLE, A CAUSE DE MON AMSTRAD. MAIS SI ON PREFERE TURBO-FORTH NO PROBLEM

REUNION F32 MI-SEPTEMBRE. LIEU: PARIS OU BANLIEUE A PRECISER. PARTICIPANTS ENVOYEZ VOS FOURCHETTES, SOIR SEMAINE DE PREFERENCE

JACCOMARD Du 27.08.88 A 11h38
- Pourquoi ne pas comprimer les fichiers en telechargement? Les arguments "contre " dans JEDI #44 sont inconsistants.
Bien sur, il faut faire gagner des sous au serveur, mais quand meme ... ARCE fonctionne sur tout compatible et est dans le domaine public.
- Il est tout a fait anormal d'ecrire des routines en Forth NON STANDARD. Dans FORUM du 21-7 on trouve asm[, handle, (out), ... qui n'ont aucun equi- valents en FIG FORTH ou 83 standard. Est-ce du Volapuck?
- Je ne suis pas sur que le Forum soit bien le lieu pour corriger des bogues

----

FORTH7 Du 28.08.88 A 18h32
Il n'y en a que pour TURBO-FORTH dans JEDI et son forum: on attend du C, du LISP, du PROLOG, du PASCAL, du MODULA du LSE, de l'APL et meme pour d'aucuns du bon FIG-FORTH au standard cacochyme Mais a qui la faute? Un Turbo-Forthien qui squatte JEDI vous le demande, tres chers inevitables grincheux...

```
\ UPC pour non anglophones
: UPC.FR ( car -- car' )
  upc " caaaaeeeeiiioouuu" 2 pick scan nip negate ?dup
  if  " CAAAAEEEEIIIOOUUU" + + C@ nip  then ;
```

convertit un caractere en majuscule, accentues compris. La premiere chaine " caaaaeee..." est a lire comme c- cedille et tous les caracteres avec accents. Dur dur ce forum sans accent dans ce cas particulier !!! Adaptable outre-Rhin naturlich. C'est en F83 standard. En FIG ? : sais plus...

SECRETAIRE Du 29.08.88 A 10h13
JE VAIS ME FACHER...
Il semble que seul F32 et deux ou trois autres connectants saisissent la finalite du FORUM.
Certes, passer une annonce ou demander une bricole passe encore, mais il ne faut pas perdre son temps a polemiquer sur le bien fonde de l'existence de SAM*JEDI. Si vous voulez faire des economies, ne vous connectez pas!
Maintenant, et je ne le repeterai plus, tout ce qui a un certain interet dans le FORUM sera publie dans JEDI au fur et a mesure de son passage sur le FORUM. Donc, si vous n'etes pas presse, attendez la parution de votre magazine favori.
Pour ceux qui veulent progresser dans leur domaine, qui ont des trucs a proposer ou des reponses a apporter, ceux qui ont envie de reunir les gens autour d'un grand projet (dixit F32...), n'hesitez pas, le FORUM est integralement disponible.
Et si pour l'instant le FORUM n'est pas divise en themes, c'est parce que nous ne sommes pas encore envahis par les messages. Pour illustration, le serveur TVSOFT (FR3 telematique) fait une moyenne de 115 connexions simultanees, avec des pointes de 500 appels dans la tranche 12h00-14h30.
Enfin, l'usage du FORUM n'est pas limite a TURBO-Forth. Comme l'exprime si bien FORTH7, parlez egalement de C, PASCAL, dBASE, etc..., et que ce ne soit pas toujours aux memes de faire de l'information.

SECRETAIRE Du 02.09.88 A 09h55
DIFFUSION DU MODULE 5 DE TURBO-Forth: Ce module 5 contient les nouveaux fichiers source de TURBO-Forth:
KERNEL.TXT et NOYAU.TXT
Ces fichiers, modifies par rapport a ceux du module M2, sont adaptes a la meta-compilation et compactage par le nouveau meta-compilateur PHENIX disponible apres compilation des fichiers suivants:
META.FTH, META-BAT.FTH et COMPACT1.FTH
La documentation de PHENIX est disponible dans le fichier: phenix.doc
Un exemple de meta-compilation avec compactage est disponible par execution des fichiers:
QUESTION.FTH et QST-BAT.FTH
Ce dernier programme genere par meta- compilation un programme .COM minimal, compacte, c'est a dire sans en-tete

Suite en page 14

hard---F32

NOTIONS SUR LA MACHINE DE PILE (STACK MACHINE)
COMMENTAIRE SUR UNE APPLICATION: LE NC4016
ADAPTATION AU LANGAGE DE PROGRAMMATION

J'imagine que tout le monde connait le microprocesseur, à savoir ce circuit de base qui execute du logiciel. Comme pour tout objet technique, comme pour les 12 mètres de la coupe de l'America ou comme pour les dinosaures, tout commence par une architecture simple mais efficace. Ensuite l'évolution arrive faisant croître performance et complexité, de pair. Survient toujours un moment où l'architecture est exploitée à fond, mais où demeure la concurrence.

Alors la voie du progrés passe par un raffinement, une surenchère technologique de plus en plus disproportionnés aux gains résultants. La fin est toujours la même: un principe nouveau émerge irrésistiblement, à la fois plus simple, plus performant et économique: plus élégant en un mot.

L'apparition récente des RISC, ou calculateurs à jeu d'instruction réduit, l'illustre bien. Un vertige a saisi les designers quant à faire mieux (plus) que les 80386, 68030 et consort. Solution adoptée: 1) réduire le jeu d'instruction du chip donc sa taille et sa complexité.
2) conserver le genre Harvard: les données et le code ont un accés séparé au processeur.
3) utiliser cette place libre par plusieurs jeux de registres, le changement de contexte reste interne au chip et l'on contourne ainsi le plus gros manque.

Un jeu d'instruction large ou réduit vous laisse aussi démuni, en effet, quand vient l'horrible chose qu'est un changement de tâche ou, pire, une interruption. Alors, tous ces beaux paramètres qui meublent les nombreux registres de votre CPU, il vous faut les emballer et ranger dans un coin, un peu comme s'il fallait totalement remeubler un appartement pour changer d'activité. Changer de tâche c'est faire cette maneuvre pour le lendemain, répondre à une interruption c'est le faire à 3 heures du matin sur alarme. Il vaut mieux être soi-même garde-meuble et déménageur: le 80386 prend cent coups d'horloge pour se retourner plus 4 par paramètre à sauver.

Les RISC ont choisi d'habiter un immeuble au lieu d'un F10 avec piscine et tennis. En contrepartie une douzaine de studios sont accessibles à l'habitant, qui passe par l'ascenceur de l'un à l'autre, chaque studio étant meublé pour une tâche. Evitez seulement les applications à 13 tâches...

On atteint tout de même de belles performances, mais à quel prix ! Plus question de programmer en assembleur, seul un compilateur de luxe peut bâtir une application qui tienne compte du contexte de 192 registres utilisés simultanément.

Ce n'est pas réellement un mal, car il fallait bien un jour s'y résoudre, mais maintenant il n'y aura tout simplement pas à choisir, c'est fait... Adieu donc aux artistes, amoureux du patch génial et du jump minimal; désormais un compilateur de 20 Moctets veille sur vos oeuvres.

Quand il s'est agi de faire le NC4000, qui n'est pas un RISC quoiqu'on en dise, la structure même du Forth a donné une autre solution: tous ces ingénieux mécanismes s'envolent dans l'espace des RETURN. Le processeur évolue dans un espace non plus linéaire comme la machine de Von Neuman, ni à deux dimensions comme un Harvard, mais tridimensionnel. Changer de contexte revient à pointer un autre plan, ce qui prend de un à 4 cycles au pire. Return prend 0 cycle.

L'utilisation est encore plus simple qu'avec les micros ordinaires qui émulent le Forth, en particulier le noyau est trés petit puisque les 40 primitives sont des paires d'octets.

Il y a donc quatre bus séparés alimentant ce chip. La mémoire principale, les piles paramètres et retour, et aussi un port I/O. Ce dernier pourrait faire partie de la mémoire, mais il faut croire que les concepteurs n'étaient plus à une vingtaine de pins prés. Il faut dire que tout ce trafic prend beaucoup de place, donc malheureusement un boitier "fakir".

Voilà que maintenant il va nous parler du boitier, en entend-je soupirer quelques-uns. Et oui, il faut savoir que cette puce, petite donc pas chère, demande un boitier de 144 pattes, et qui coûte 1200 F. On reviendra sur ceci car c' est une première raison de l'échec (et oui) du NC4016.
Bref, il y a donc un port I/O, mais quel port! le plus sioux qui soit. On peut lire, écrire, mémoriser en entrée ou en sortie, guetter un niveau sur n'importe lequel des 16 bits avec en plus contrôle de polarité et masquages. Si avec çà on n'entend pas "papa maman" en baissant l'oreille, c'est juste faute de l'avoir programmé. De fait en ambiance de contrôle de process, le NC4016 jongle avec ses trois bus de données et ne les ralentit pas du tout en besognant coté I/O, oú il fait mieux de loin que n'importe quel autre microcontroleur.

MACHINE DE PILE

Si je traduis "stack machine" par machine à pile, vous penserez "chouette, en plus on peut l'emmener à la plage !".
Circulez, çà n'a rien à voir.
Les deux mémoires Return et Param sont des Ram tout ce qu'il y a d'ordinaire, mais le fonctionnement préféré est de les utiliser comme des piles, ce qui est trivial en Forth. A cet effet les pointeurs d'adresses incrémentent/décrémentent tous seuls à l'aide d'additionneurs dédiés. Le générateur d' adresse de la mémoire principale auto-incrémente aussi, mais c'est là tout à fait banal.
En quoi réside l'intérêt d'une telle machine ?
- les piles dégagent le processeur de toute la surcharge de gestion du contexte, lui laissant la partie noble de l'éxécutable, c'est à dire ces instructions qui vous intéressent. Mesurer la performance en MIPS (Millions d'Instr./S) a alors un sens.
- en contrepartie le processeur devient trés gourmand en débit mémoire, car digérant vite il revient à la charge dès le cycle suivant dans la plupart des cas. Quelle est la fonction du microprocesseur ? Bouffer du code. Celui-ci et les RISC sont capables de consommer tout ce que des mémoires rapides peuvent fournir, et plus.
- comme on a trois accés simultanés (les I/O en effet sont généralement moins demandeurs), le débit brut est amélioré d'autant. Ceci recouvre en partie le premier point.

L'éxécution du Forth N'EST PAS un avantage du point de vue de la performance. Un code en C fera aussi bien s'il use autant des piles. Tempérons tout de même un peu: il sera plus facile de faire du bon code "à pile" en Forth. Mais si vous voulez vendre un circuit, les utilisateurs voudront du C. les deux cas demandent une machine identique.

Une remarque au sujet de la performance. S'il faut des Rams rapides pour alimenter ce circuit, le système devient cher. Une carte Cpu coûte facilement une dizaine de KF rien qu'en composants si l'on installe quelques megamots de Ram.
Pas de panique! En prenant de la mémoire plus lente et nettement moins chère, on gagne trés vite sur les coûts, le système restera tout de même impressionant avec des temps d' accés de 100 à 120 nS qui sont économiques. L'important est de tailler sur mesure le rapport prix/performance, sachant qu'on en a toujours "sous le pied" pour faire mieux.

Ecrire un noyau Forth pour le chip F32 doit être aisé, un portage du C un peu moins, mais C n'est pas machine-dépendant, c'est à dire que son noyau matériel est minimal.

à suivre

soft---F32

ENVIRONNEMENT HIERARCHISE
PRIMITIVES ASSOCIEES

Quel est le plus compliqué, la forêt ou l'arbre ? La pomme ou le pépin ? Ni l'un ni l'autre. C'est nous qui avons l'idée de complexité. Quand on s'occupe de la forêt le pépin ne compte pas. Les objets accessibles à nos manipulations ne sont pas emboités mais hiérarchisés. A chaque niveau de notre scrutation, l'objet a à peu prés la même complexité, celle que nous lui attribuons.

Les approches linéaire et hiérarchisée sont deux modes de représentation, l'un primitif(rappelez-vous la peinture avant la perspective), l'autre adapté à la dynamique du réel.
a,b,c,d...,z sont divers objets comparables. a contient aa,ab,ac... et aa juste aaa, alors que ab est: aba,abb,abc..
On écrit (a(aa(aaa),ab(aba,abc(abca),abd, et la liste finit par zzzzzy,zzzzzz)))))).
L'objet Toto possède une adresse [jyrdttrd] qui définit une profondeur de 8 par rapport au point d'entrée. On y accède successivement au travers du 10eme puis 25e, 18e, 4e, 20e, 20e, 4eme élement rencontré à un niveau donné.

La liste à plat ci-dessus, comme celles qu'utilise LISP doit donc être parcourue avec à la main le chemin spécifié,et on arrive au but. De quoi est-elle faite ?

Le pire cas est que chaque élément soit lui-même étalé, toute la base de donnée doit être lue lettre par lettre. Plus souvent on a recourt à des indirections. La liste contient juste un pointeur sur l'objet recherché, et des séparateurs structurants.

Les micros évoluent dans un espace linéaire d'adresses physiques. Dans les OS modernes et certaines applications de gestion de base de données, la machine logicielle évolue dans cet espace virtuel hiérarchisé. Bien.
Reste plus qu'à donner l'adressage hiérarchisé au micro et on pourra commencer à travailler sérieusement. Programmer dans cet espace doit être transparent jusqu'au niveau le plus bas possible.
********Dans ce type d'adressage, l'adresse absolue est celle du descripteur de chemin, le path.

```
        A [Toto] on trouve        [Toto]---->     10
                                                  25
                                                  18
                                                   4
                                                  20
                                                  20
                                                   4
```

puis un bouchon quelconque pour cesser de ramer.

Une fois cette liste empilée dans les paramètres, le μ part du début de la liste à plat, et avale son contenu qu'il filtre à grande vitesse. Quand on trouve (le niveau courant NIV est incrémenté, contraire pour) . Quand NIV atteint 0 c' est qu'on a rencontré la fin de liste. Au cours de l'excursion le path courant est comparé au path de consigne. Par exemple si on est hors path aprés un (pas la peine de traiter le flot en détail, il suffit de courir au prochain) de même niveau, et reprendre là.

Quelques pointeurs sont à stocker en cours de route, en particulier ceux du voisinage de la destination, la localité l'application étant généralement réalisée.
********Dans ce contexte, l'adressage relatif consiste en un déplacement à partir du point courant.

Le chip aura évidemment en mode natif un adressage direct, et c'est l'une des raisons principales à l'utilisation de 32 bits. Seulement il faudrait que les primitives d'adres-

sage virtuel décrit plus haut se trouve le plus "bas" possible l'idéal étant d'en cabler tout ou partie.

Je considère la mise au point de cette fonctionnalité en virtuel sur Turbo-Forth comme essentielle pour F32.

à suivre

hard---F32

JEU D'INSTRUCTION
IMPLEMENTATION

Supposons que nous ayons un Turbo-Forth muni de toutes subtilités pour manier des mots de taille quelconque. Dans un dictionnaire F32 nous avons un noyau complet pour exécuter n'importe quelle application.

Supposons que Sam*jedi nous permette de travailler à plusieurs. Les taches à accomplir sont les suivantes.

L'équipe langage vérifie la compatibilité du noyau F32 en émulant avec les applications existantes de Turbo-Forth. Elle intègre par ailleurs les primitives spéciales F32.

Les candidats à cette équipe sont encouragés à étendre leurs attributions. Une équipe commence à une personne. Une personne peut faire partie de deux équipes, de préférence adjacentes. (J'espère que les n équipes ne seront pas faites de mon unique personne).

L'équipe noyau constitue un noyau(pardi!) qui soit:
- minimal
- à largeur de mot non fixée sur 16 bits
- compatible Turbo-Forth: vérifié en émulant les applics existantes
- avec l'équipe VLSI négocier les primitives à câbler ou non(simplicité, vitesse, etc)
- à niveau de définition réglable: chaque primitive aura un paramètre associé autorisant à l'éxécution soit:
  - simple fonctionnalité virtuelle
  - idem plus timings puce virtuelle
  - idem plus timings système virtuel
  - idem plus activation interne du chip

Les candidats...

L'équipe VLSI communique au noyau les fonctionnalités en service. Si on me dit: il faudrait un SWAP(possible...) je regarde tout de suite si N et T peuvent permuter. (sinon, on n'est pas dans la...)! Elle informe le noyau des détails de chaque primitive, en particulier des timings, pour permettre au langage de faire des simulations de temps réel. G. Dumur appréciera la puissance et la simplicité de ce simulateur.

Inscrivez vous en vitesse.

Suite de la page 12

et contenant exclusivement les routines necessaires a son fonctionnement.
Le module M5 peut etre commande a l'association contre 10 timbres a 3,70 Fr.
Prix des modules M4+M5: 70,00 Fr. Pas de cheque en dessous de cette somme, SVP. Envoyer votre commande au siège de l'association.

SECRETAIRE Du 06.09.88 A 10h48
QUI EST AU COURANT DE L'EXISTENCE D'UN PROGRAMME DE CONVERSION DES FICHIERS dBASE EN C...?
INFO: dBASE IV pas dispo avant nov88. Possibilite reprise de votre dBASE III par LCE si achat avant fev88 pour moins de 1000 fr.

F32 Du 06.09.88 A 17h36
LA REUNION F32 SE TIENDRA A PARIS LA DERNIERE SEMAINE DE SEPTEMBRE. 69204590 POUR NEGOCIER LE JOUR SI VOUS N'ETES PAS ENCORE INSCRIT. F32 DEMARRE AVEC TROIS NIVEAUX D'ACTIVITE:
- LANGAGE: 32 BITS POUR TURBO-FORTH/EXECUTION DES APPLICS EXISTANTES AVEC LE NOYAU F32 ET CORRECTIONS/BENCHMARKS
- NOYAU: DEFINITION/MINIMALISATION/COMPACTAGE DES OPCODES/TIMINGS
- VLSI: INTEGRATION DU NOYAU/FEEDBACK VERS EQUIPE NOYAU
IL Y A DE LA PLACE POUR TOUT LE MONDE VOIR PLUS DE DETAILS DANS JEDI PROCHAIN (Ndlr: dans présent n°)
INSCRIVEZ-VOUS DANS F32
POURRAIT-ON CREER UN SOUS-FORUM F32 AVEC LE MOT DE PASSE DE F32?

FORTH-83

FORGETTABLE
INTERNAL NAMES

MICHAEL HORE - NUMBULWAR, AUSTRALIA

From time to time, there have been various proposals for dealing with internal, or local, names. By these, I mean the names of words that are only used internally in an application, never called from outside. A particularly nice scheme was presented by Dewey Val Schorre, back in *Forth Dimensions* II/5 ("Structured Programming by Adding Modules to Forth"). Dewey brought the notion of a "module," and presented three words (**INTERNAL**, **EXTERNAL**, and **MODULE**) to implement this notion. Each module is a self-contained unit, and communicates with the outside world by means of words defined between **EXTERNAL** and **MODULE** (as well as any constants or variables defined before **INTERNAL**). Words defined between **INTERNAL** and **EXTERNAL** may be referenced from within the module, but are not accessible from outside. Each module may be a complete application or a building block in a larger application.

Dewey's implementation of these words was simple but effective, involving the replacement of a dictionary link to remove the internal words from the search chain. Dewey acknowledges that such a simple implementation does not allow the dictionary space savings that could be possible — the headers of the internal words are unused after **MODULE** has been reached, but remain in the dictionary. (Note that the parameter fields of these definitions are needed for execution, but not the headers.)

Perhaps one reason why Dewey's words have not gained a wider acceptance is that, since his article, more elaborate vocabulary structures have evolved, and the effect of Dewey's words can be obtained by using vocabularies in an appropriate way. I have always felt, though, that Dewey's words state more clearly what is going on, since vocabularies have many other uses besides this particular one.

At this point, I am not going to attempt a full justification of a modular style of programming, which is now accepted as an essential discipline in the construction of any significant application. Let me just say that I have found it very useful to have this discipline enforced (the temptation to lapse into spaghetti code is always there). If I try to call an internal name from outside a module, it is good to have the system throw it back in my face. It means I probably have not understood

Suite page 16

STRUCTURE DE DONNES PAR ENREGISTREMENT
et
INITIALISATION D'IMPRIMANTE PAR MENU

par Yves SURREL

Diffusion: module M6 de TURBO-Forth et
téléchargement 3615 SAM*JEDI

Voici deux courts programmes commentés traitant d'une structure de données extrêmement pratique, l'ENREGISTREMENT. Je sais qu'elle a déjà été implémentée en FORTH, mais permettez-moi néanmoins de vous soumettre ma version. Vous trouverez dans le fichier RECORD.FTH toutes les informations nécessaires.

Le deuxième programme (fichier IMPRIM.FTH) est un exemple d'application: un programme de configuration rudimentaire d'imprimante par menu. On se déplace dans le menu par les flèches verticales et les touches HOME et END. La sélection d'une option se fait en appuyant sur la barre d'espace. On sort en appuyant sur RETURN. Les attributs d'affichage sont prévus pour une vidéo couleur. Les modifier si nécessaire. Pour un résultat parfait, rajouter 2 ATTR ! dans la boucle du mot TICTAC (fichier TIME.FTH).

Une remarque pour ceux qui doutent encore de l'intérêt d'avoir abandonné les blocs: essayez d'imaginer l'aspect du mot CONFIGURE (dans IMPRIM.FTH) tronçonné en blocs...(IF...THEN imbriqué dans DO...LOOP imbriqué dans IF...ELSE...THEN imbriqué dans CASE...OF...ENDOF imbriqué dans BEGIN...UNTIL!).

LISTING: RECORD.FTH

```
ONLY FORTH DEFINITIONS DECIMAL

0 CONSTANT 'TAB     \ Adresse début du tableau
0 CONSTANT <ENR>    \ Longueur de l'enregistrement
0 CONSTANT #ENR     \ Nb max d'enregistrements
                    \ dans un tableau

VARIABLE ^          \ Index variant de 0 à #ENR-1

: ^! ( n -- ) ^ ! ;

: CHAMP   (S offset_depuis_debut_enreg <nom du champ> -- )
  CREATE ,
  DOES>   (S -- add )  \ L'enreg. courant est pointé par ^
  @                    \ Offset depuis debut enregistrement
  ^ @ <ENR> * +        \ + la longueur des enreg.
                       \ précédents
  'TAB + ;             \ + adresse début du tableau

\ Le mot suivant est assez complexe. La syntaxe
\ d'utilisation par contre est simple:
\ ENR: BONHOMME 15 NOM 15 PRENOM 2 AGE 16 TELEPHONE ;
\ où l'on définit l'enregistrement BONHOMME comportant
\ 4 champs de longueurs 15, 15, 2, 16 respectivement.
\ On définit donc 5 nouveaux mots en une seule ligne.
\ Après compilation du nom du champ par le mot CONSTANT,
\ on lit le flot d'entrée en s'attendant à trouver
\ alternivement les longueurs des champs et leurs noms.
\ La fin est donnée par le point-virgule qui est ici un
\ séparateur et non un mot.

: ENR:       ( mot de definition )
  0 CONSTANT     \ Compilation nom de l'enreg.
  HERE 2- 0         (S -- pfa 0 )
  \ On garde le pfa sur la pile pour l'initialiser
  \ une fois que la longueur totale de l'enregistrement
  \ sera calculée. 0 est la longueur initiale.
  BEGIN   (S pfa lgenr )
         \ Lire le flot d'entrée...
         SOURCE DROP >IN @ + C@
         \ Prochain caractère à lire
         ASCII ; = NOT  (S pfa lgenr )
  WHILE
         \ Tant que l'on n'a pas rencontré le
         \ point-virgule...
         \ ... on saisit la longueur du prochain champ
         \ à compiler
         BL WORD NUMBER DROP  (S pfa lgenr lgchamp )
         SWAP                 (S pfa lgchamp lgenr )
         \ La longueur actuelle de l'enregistrement
         \ est donc l'offset du prochain champ par
         \ rapport au début de l'enregistrement.
         DUP CHAMP  (S pfa lgchamp lgenr )
         \ On compile le nom du champ
         +  (S pfa lgenr' )
         \ Nouvelle longueur de l'enreg.
  REPEAT
  SWAP !  \ Le nom de l'enregistrement fournira
          \ à l'éxécution sa longueur
  1 >IN +!   \ On saute le ; dans le flot
  ;

\ Remarque importante: il faut que la définition d'un
\ enregistrement tienne sur une seule ligne!

\ Le mot suivant permet de définir une structure
\ de tableau d'enregistrements.
\ Syntaxe:
\ 27 BONHOMME TABLEAU BONSHOMMES
\ définit une zone mémoire de 27 enregistrements
\ de structure BONHOMME

: TABLEAU            (S nb lgenr -- )
  CREATE 2DUP
  ,    \ pfa contiendra la longueur d'un enregistrement
  ,                     \ pfa+2 le nombre d'enreg.
  * ALLOT               \ pfa+4 --> données
  DOES>   (S compil: -- pfa ; exec: -- )
  \ L'éxécution du mot (par exemple BONSHOMMES) est
  \ indispensable pour initialiser les constantes <ENR>,
  \ #ENR, 'TAB et la variable ^.
  \ Ceci est équivalent à l'utilisation de WITH
  \ en PASCAL).
  DUP @ IS <ENR>      (S pfa )
  DUP 2+ @ IS #ENR    (S pfa )
  4 + IS 'TAB         (S -- )
  0 ^! ;

: ^>      (S -- )         \ Pointe sur l'enregistrement
                          \ suivant si possible.
  ^ @ 1+ #ENR 1- MIN ^! ;

: <^      (S -- )         \ Pointe sur l'enregistrement
                          \ précédent si possible.
  ^ @ 1- 0 MAX ^! ;
\ Exemple:
\ 10 ^! pointe sur le 11ème enregistrement (n° 10)
\ NOM laisse l'adresse des données correspondantes
EOF
```

Fichier: IMPRIM.FTH

```
ONLY FORTH DEFINITIONS DECIMAL
INCLUDE RECORD.FTH
INCLUDE TIME.FTH ( diffusé dans module M3 de TURBO-FORTH)
                 ( ou téléchargement 3615 SAM*JEDI       )

\ Définition de l'enregistrement. POSITION contient
\ le n° de cellule écran (ligne*80 + colonne) où
\ sera affiché la chaîne de caractères CHAINE.
\ CODES contient les octets à envoyer à l'imprimante.

ENR: CONFIGURATION 2 POSITION 25 CHAINE 5 CODES ;

25 CONFIGURATION TABLEAU CONFIGURATIONS

\ Initialisation du tableau CONFIGURATIONS

: INIT-ENR ( col lig add len c1 c2 c3 c4 c5 -- )
 0 4
 DO
       CODES I + C! -1
 +LOOP
 CHAINE PLACE 80 * + POSITION ! ^> ;
```

```
CONFIGURATIONS                  \ "Activation" du tableau.

10 4 " Initialisation" 27 64 255 255 255 INIT-ENR
10 5 " Mini:                OUI" 27 77 255 255 255 INIT-ENR
50 5 " NON" 27 80 255 255 255 INIT-ENR
10 6 " Italique:            OUI" 27 109 0 27 52 INIT-ENR
50 6 " NON" 27 53 27 109 2 INIT-ENR
10 7 " Souligné:            OUI" 27 45 1 255 255 INIT-ENR
50 7 " NON" 27 45 0 255 255 INIT-ENR
10 8 " Double largeur:      OUI" 14 255 255 255 255 INIT-ENR
50 8 " NON" 20 255 255 255 255 INIT-ENR
10 9 " Condensé:            OUI" 15 255 255 255 255 INIT-ENR
50 9 " NON" 18 255 255 255 255 INIT-ENR
10 10 " Qualité courrier:  OUI" 27 120 1 255 255 INIT-ENR
50 10 " NON" 27 120 0 255 255 INIT-ENR
10 11 " Double frappe:     OUI" 27 71 255 255 255 INIT-ENR
50 11 " NON" 27 72 255 255 255 INIT-ENR
10 12 " Proportionnel:     OUI" 27 112 1 255 255 INIT-EN
50 12 " NON" 27 112 0 255 255 INIT-ENR
10 13 " Gras:              OUI" 27 69 255 255 255 INIT-ENR
50 13 " NON" 27 70 255 255 255 INIT-ENR
10 14 " Petit interligne:  OUI" 27 48 255 255 255 INIT-ENR
50 14 " NON" 27 1 255 255 255 INIT-ENR
10 15 " Table EPSON" 27 109 0 255 255 INIT-ENR
10 16 " Table IBM" 27 109 2 255 255 INIT-ENR
10 17 " Marge à gauche" 27 108 20 255 255 INIT-ENR
10 18 " Chaîne de caractères" 255 255 255 255 255 INIT-ENR
FORGET INIT-ENR

\ Affichage de la chaine de caractères CHAINE
\ à la position POSITION

: .ITEM (S att -- )
  ATTR ! CHAINE COUNT POSITION @ $>SCREEN ;

: SELECTED (S -- )  \ Affichage jaune sur bleu, intense
  30 .ITEM ;

: UNSELECTED (S -- ) \ Affichage cyan
  3 .ITEM ;

\ Mot utilisateur. On utilise pour l'affichage le
\ mot $>SCREEN défini dans TIME.FTH pour des raisons de
\ rapidité.

: CONFIGURE  (S -- )
\ Affichage de la page-écran
  DARK 14 ATTR !
  " Selection: <ESPACE>  ---  Quitter: <RETURN>"
  1938 $>SCREEN
  CONFIGURATIONS
  30 ATTR !
  " CONFIGURATION IMPRIMANTE" 25 $>SCREEN
  SELECTED
  #ENR 1
  DO
         ^> UNSELECTED
  LOOP CR
  0 ^!

\ Boucle d'attente

  BEGIN
     FALSE   \ Drapeau de sortie par défaut
     KEY
     CASE
          0 OF    \ Appui d'une touche de fonction
               KEY   \ 2ème code envoyé
               CASE
                    72 OF UNSELECTED     (^     SELECTED
                       ENDOF \ Fl. haut
                    80 OF UNSELECTED      ^>     SELECTED
                       ENDOF \ Fl. bas
                    71 OF UNSELECTED    0 ^!     SELECTED
                       ENDOF \ Home
                    79 OF UNSELECTED  #ENR 1- ^!  SELECTED
                       ENDOF \ End
               ENDCASE
          ENDOF
          13 OF (S false ) NOT ENDOF
          \ Appui de <RETURN> => sortie
          32 OF
          \ Appui <ESPACE> => sélection
                    ^ @ #ENR 1- =
                    \ Si c'est le dernier choix...
                    IF  ( envoi chaine de caractères)
                        0 20 AT ATTOFF
                        PRINTING ON PAD 255 EXPECT CR
                        PRINTING OFF
                        0 20 AT 80 SPACES
                    ELSE        \ sinon envoyer les codes
                        5 0     \ de contrôle
                        DO CODES I + C@ DUP 255 =
                           IF DROP LEAVE
                           THEN (PRINT)
                        LOOP
                    THEN
               ENDOF
          ENDCASE
   UNTIL ATTOFF DARK ;

EOF  ( <== supprimer EOF...si:)

\ Compiler les lignes suivantes pour faire un fichier
\ exécutable appelé I.COM

: C MULTI TICTAC WAKE CONFIGURE BYE ;
' C IS BOOT
SAVE-SYSTEM I.COM
EOF
```

Suite de la page 14

either the problem or my "solution" to it! It also means that if I want to change the specification of an internal word, I know I don't have to look very far to find all the references to it.

Let us now turn to the question of the potential space savings. Note that we are not realizing this with modules implemented by means of vocabularies, any more than we could with Dewey's implementation of his words. That is the reason for the code presented here. I have found that space savings on the order of 20% is possible — the shorter the definitions, the greater the savings. This means that good Forth programmers will benefit more from this code than bad ones (bad programmers, please stop reading). This code will do even more — it will allow a whole module, parameter fields and all, to be loaded temporarily and then dismissed from memory, while leaving subsequent definitions intact. It is very useful to be able to do this with an assembler, as one obvious example.

It turns out that Dewey's three words are more suitable here than if we tried to do the job with vocabularies. That would not be impossible, just more difficult. Moreover, Dewey's words are very clear and say exactly what they mean — unlike, say, **FORGET-VOC.** Perhaps they may find a new lease on life with this implementation.

Working in this area inevitably brings us to various implementation dependencies, since the Forth standard quite rightly leaves unspecified such details as the internal structure of the dictionary. So what we will do is focus on one specific

Suite page 26

La compilation en FORTH standard de tout programme se fait à partir d'un source en code ASCII. Trois inconvénients sont évidents:

- le temps de compilation est important puisque le compilateur doit traduire l'ASCII en code exécutable avant de le placer en mémoire;

- la taille du source est importante (UTILITY.BLK occupe 112k sur le disque);

- il est difficile dans ces conditions de séparer du noyau les utilitaires, pourtant devenus inutiles après mise au point et compilation.

Une solution commode est de sauvegarder sur disque le code compilé, par exemple la fraction de dictionnaire correspondant au débogueur, et de le recharger directement lorsque le besoin en survient. Si la première partie est facile (SAVE ne fait pas autre chose), la seconde est plus délicate: comment rétablir les liens avec le dictionnaire existant? Une solution est proposée dans cet article, avec des exemples d'application.

Le programme est constitué en toute logique des deux parties 'sauvegarde' et 'rechargement', ceci afin de permettre la sauvegarde de SAVBIN sous forme binaire.

Voyons d'abord la sauvegarde, qui conditionne le reste.

SAVBIN : écrans 12 à 20.

Le fichier binaire sera rechargé en un emplacement mémoire différent de celui qu'il occupait à la sauvegarde. De même est probablement différente la structure du dictionnaire à ce moment. Il faut donc:

- conserver l'adresse de compilation d'origine, pour la comparer à l'adresse actuelle de rechargement: la différence sera ajoutée aux adresses modifiées, et seulement à celles-ci (celles des mots antérieurs au segment sauvegardé sont évidemment inchangées);

- sauver aussi les liens avec le dictionnaire: ceux le liant au début et ceux le liant à la fin du segment.

Quelles adresses seront à recalculer? Tout simplement, celles qui sont modifiées selon l'emplacement mémoire du programme. Faisons donc DEUX compilations successives, et regardons ce qui a changé: tout octet modifié entre les deux versions appartient en principe à une adresse qui devra être recalculée. Une table est construite, TAB_BIT, qui contient ces informations: chaque octet du programme est représenté par un bit de la table, ce bit est mis à 1 lorsque les deux octets correspondants des deux versions compilées sont différents. La table est ainsi 8 fois plus petite que le programme sauvegardé.

Il faut aussi sauver les 8 liens avec le dictionnaire (2 pour chaque 'brin' d'un vocabulaire: un en 'haut' du segment, l'autre en 'bas'). En fait, on sauve l'écart entre les LFA's et le début du programme. Ceci se fait en deux phases: les liens 'supérieurs' (ceux par lesquels commence toute recherche dans le vocabulaire) sont contenus dans CONTEXT, donc directement accessibles.

Pour déterminer les liens 'inférieurs' (qui lient le programme au reste du dictionnaire), on suit les brins un par un depuis le 'haut' du segment jusqu'à une limite inférieure, qui est fournie par le LFA du premier mot sauvegardé, début de la seconde version compilée.

Il ne reste plus qu'à placer tout cela dans une zone 'paramètres', située juste au-dessus de la table précédente, puis à écrire sur disque l'ensemble "version2 + table + paramètres" .

SBIN (écran #20) fait tout cela: il exige deux paramètres, le nom du premier mot du programme et le nom du fichier binaire qui sera écrit:

SBIN <1er_mot> <nom_fichier>

Notons que le <1er_mot> doit être connu ou déterminé par le programmeur. Il sera d'autre part utilisé pour 'effacer' le programme devenu inutile. Ce premier mot doit donc être 'remarquable' et facilement mémorisé. Voyez par exemple l'écran #12, où le programme SAVBIN est 'marqué' par le mot : *SAVBIN* ; défini en tout début, et suivi du 'mot d'effacement' SAVEND. Les écrans #9 et 10 donnent des exemples de mots d'appel de programmes binaires.

Une extension .BIN est automatiquement ajoutée au nom du fichier créé.

SBIN commence par compiler une première version du fichier OUVERT à ce moment: donc faire précéder IMPERATIVEMENT toute l'opération par

OPEN <src_pgm_à_compiler>

Pour simplifier, on suppose ici que ce programme est écrit en syntaxe F83 rigoureuse, c'est à dire sous forme d'écrans-blocs, et que l'écran #1 est un écran de chargement. Un simple 'OK' (cf. LOC, #17) fait alors tout le travail. Les inconditionnels de TURBO-FORTH devront modifier cela ...

LOC en profite pour calculer les CFA du <1er_mot> dans les deux versions, leur différence donne la taille du programme, puis il calcule les 'offsets' des LFA inférieurs (leur distance par rapport à la base).

Remarquez la troisième condition ABORT dans LOC: lorsque la longueur du programme est multiple de 256 (100 hexa) l'octet bas des adresses dans les deux versions est le même, leur 'image' dans TAB_BIT sera à 0, et RELOC de LODBIN (cf infra) modifiera la partie haute de l'adresse ... et l'octet qui suit. C'est le crash. Ajoutons donc un NOOP ou un 1 ALLOT au programme source.

SBIN ouvre ensuite un fichier binaire sur disque, calcule la longueur du programme, construit TAB_BIT (cf. #13) et la zone paramètres (#18), enfin écrit le tout, depuis le début de la version 2 (le champ VIEW du premier mot) sur le disque, ferme le fichier, et vous annonce qu'il a terminé.

LODBIN : écrans 2 à 8.

SAVBIN utilisant certaines variables de LODBIN, ce dernier doit être compilé en premier.

LBIN (écran #8) demande un seul paramètre: le nom du fichier binaire à charger, sans extension; un nom de lecteur et un chemin d'accès peuvent être donnés, ceci est DOS-standard.

Ce fichier est chargé en HERE (actuel) par (LBIN), qui vérifie sa validité simplement en comparant l'adresse sauvegardée de TAB_BIT avec celle actuellement en mémoire. Si vous avez modifié votre noyau dans l'intervalle, votre fichier sera rejeté comme non valable ...

Puis (LBIN) (écran #6) va chercher les paramètres ainsi chargés et les utilise pour 'reloger' les adresses, après avoir déplacé DP, et rétablir les liens avec le dictionnaire.

Quelques précisions sur les mots MASK et BIT_1? de l'écran 2. MASK est un tableau contenant les nombres hexa 01, 02, 04, ... 80: il permet d'examiner le bit 0, 1, 2, ... 8 d'un octet. BIT_1? l'utilise pour empiler un drapeau vrai si ce bit est à 1. Notez le code directement entré 74 C, 3 C, correspondant à JZ +3 et qui évite de créer une étiquette.

Comparaisons :

Elles sont faites sur le même programme:

- le source DECOMP.BLK occupe 27648 octets sur le disque et se compile en 26 secondes;

- DECOMPIL.BIN : 5248 octets, 4 secondes;

- pour TURBO-FORTH, DECOMP.FTH occupe 12017 octets et se compile en 45 secondes (ECHO OFF, sinon 1 min 28 sec), sans parler du temps passé à adapter le programme ...

Les commentaires sont superflus.

Utilisation.

Les écrans 9 et 10 contiennent deux exemples d'appel direct d'un fichier binaire.

Le premier a été obtenu à partir d'un décompilateur (une version élaborée de celui intégré dans F83). Avant de le sauvegarder, *DECOMP* et DECOMPEND ont été ajoutés en tête du source, respectivement marque et effacement du segment de dictionnaire contenant le programme. Ces mots sont une facilité, et non une obligation: le programme fonctionne sans eux. Ce source ne nécessitait aucune autre modification, mais ce n'est pas toujours le cas, voir plus loin.

Ensuite

OPEN DECOMP SAVBIN *DECOMP* DECOMPIL

ont ouvert, chargé et sauvé le décompilateur.

Pour voir si 'celà marche', faire

DECOMPEND 2 TIMES

(pour effacer les deux versions) puis

DECOMPILATEUR <Enter> : le voilà prêt à l'emploi.

A la fin, DECOMPEND suffit pour l'ôter du dictionnaire (avec tout ce qui le suit ...).

Le second exemple permet la conversion de SAVBIN: en effet, LODBIN doit évidemment être présent dans le noyau Forth, mais SAVBIN n'a aucune raison d'être là en permanence.

SAVBIN a nécessité une adaptation, avant sa conversion en fichier binaire. Dans l'écran #12, la marque *SAVBIN* et 'l'effaceur' ont été ajoutés. Notons que les définitions sont placées dans le vocabulaire DOS, ce qui ne pose pas de problème particulier pour le rechargement. Mais si le programme crée un vocabulaire, celui-ci sera bien rechargé, mais son nom n'apparaîtra pas dans VOCS, bien qu'il soit présent dans le dictionnaire, et ses liens avec les autres vocabulaires risquent d'être perturbés. Attention donc: il est peut-être préférable de tout mettre dans le vocabulaire principal, qui est bien la place d'un programme éphémère.

Un autre point fait toucher du doigt les subtilités de la conversion binaire: dans le mot 2**N (écran #13), le code 1PUSH doit être remplacé par son expression complète. En effet, ce code ASSEMBLER empile le contenu du registre AX puis effectue un saut RELATIF à la routine NEXT d'exécution du mot suivant. Or cette valeur est bien différente pour les deux versions compilées, et RELOC la considérera comme une adresse ABSOLUE, à laquelle il ajoutera le décalage du programme: mais NEXT n'a pas changé d'adresse, alors le système se plante ... Le plus simple est ici de 'recopier' la routine NEXT, on y perd seulement quelques octets.

Après SBIN *SAVBIN* SAVBIN, on obtient un fichier binaire SAVBIN.BIN sur le disque, et 3 versions dans le dictionnaire, qu'il faut éliminer par SAVEND 3 TIMES avant de le recharger par SAVBIN, tout simplement.

Deux points importants restent à vérifier: les 'phrases' exécutables du programme source, et les vectorisations. Pour les premières, un chargement binaire ne pourra pas en général les exécuter, sauf si elles portent sur le programme lui-même (par exemple une initialisation de variables 'locales'). Les secondes posent un problème identique. Une solution consistent à placer ces initialisations et vectorisations dans le dernier mot du programme, puis à créer un mot d'appel comme celui de l'exemple de l'écran #7, lequel justement exécute le dernier mot compilé du programme.

Une utilisation générale et tout à fait rationnelle de ce programme serait la reconstruction du noyau F83, dans lequel débogueur, éditeur, assembleur et décompilateur seraient enlevés et placés dans une bibliothèque 'binaire'.

JEDI proposera-t-il cette nouvelle version en téléchargement?

A. Jaccomard, Juillet 88.

écran #0.

```
                 ******* LODBIN ******
```

Ce programme permet le chargement d'un fichier binaire, qui aura été préalablement construit et écrit sur disque (à partir d'un fichier source .BLK), à l'aide du programme SAVBIN.

Son chargement se fait par

LBIN (nom)

l'extension .BIN est rajoutée par le programme.

Il se place à partir de HERE, les liens avec le vocabulaire étant recréés.

Il est ainsi possible de construire puis recharger des programmes ASSEMBLEUR, EDITEUR, DEBOGUEUR, DECOMPILATEUR, ..., en libérant le noyau de fonctions utilisées uniquement en compilation ou mise au point, mais inutiles lors de l'exécution.

écran #1.

```
\ LODBIN : chargement.                              02Jui88JaD

ONLY  FORTH ALSO DOS ALSO  DEFINITIONS

   2 10 THRU

CR CR  BRILL
.( LODBIN est chargé. Pour l'utiliser, faire:) CR CR
.(    LBIN (nom_fich) ) CR CR
.(          l'extension .BIN est implicite.)
CR CR NORM

ONLY  FORTH ALSO  DEFINITIONS
```

écran #2.

```
\ LODBIN : MASK, BIT_1?                              02Jui88JaD
HEX
CREATE MASK 201 , 804 , 2010 , 8040 , \ oct. 01, 02, ... 80.

CODE BIT_1?   (S offset,adr_déb -- f : vrai si bit=1 )
  BX POP  CX POP  CX AX MOV  AX SHR  AX SHR  AX SHR  AX BX ADD
  0 [BX] AL MOV  7 # CX AND  MASK # BX MOV  CX BX ADD
  AH AH SUB  0 [BX] AL AND  74 C, 3 C,  -1 # AX MOV  1PUSH C;
DECIMAL

VARIABLE TAB_BIT            \ adr table de conversion d'adresses.
VARIABLE DECAL              \ contient le décalage.

\S MASK masque le bit 0, 1, 2, ... , 7 d'un octet.
BIT_1? teste un bit dans une table, et empile un drapeau vrai
s'il est à 1.
```

écran #3.

```
\ LODBIN : ouvre et lit un fichier sur disque.   02Jui88JaD

: OPEN-BIN          (S adr_fcb -- )
   CR  BRILL ." Un instant ... "  CR
   DUP 15 BDOS              \ ouverture du fichier sur disque
   DOS-ERR? ABORT"  Fichier BIN non trouvé."   NORM
   DUP  16 + @  SWAP  14 + !  ;  \ place lgr enrg. ds FCB

: READ-BIN          (S fcb,ad_dest -- ad_parms)
   DUP 26 BDOS  DROP              \ place la DTA en ad_dest
   OVER  16 +  @  DUP             \ lgr du fichier
   SP@ SWAP - 100 - HERE U<       \ assez de place?
   CR INV ABORT" Mémoire disponible insuffisante." NORM
   + 128 - SWAP  20 BDOS DROP ;   \ lecture séquentielle
```

écran #4

```
\ LODBIN : FICH>FCB, RELOC.                          02Jui88JaD

: FICH>FCB   (S -- )
   BL WORD COUNT          \ lit un nom de fichier,
   FCB1  (!FCB)           \ et le place dans le FCB.
   " BIN"   FCB1  9 +  SWAP  CMOVE ;

: RELOC            (S adr_base,lgr -- adr_base )
   0 DO            \ pour chaque octet du fichier
     I  TAB_BIT @  BIT_1?
     IF  DUP  I + DECAL @    \ vrai = adr. modifiée,
         SWAP  +!  2          \ le décalage y est ajouté.
     ELSE  1  THEN             \ faux = octet suivant
   +LOOP  ;

\S RELOC rétablit les adresses du programme binaire.
```

écran #5.

```
\ LODBIN : PARMS, RELINK.                            02Jui88JaD
: PARMS          (S base,adparms --  adparms,base,lgr )
   2DUP @ + ( base+lgr )  TAB_BIT !      \ début de la table
   2DUP 2+ @ - ( décalage ) DECAL !
   DUP @ SWAP -ROT  ; \ place 2 val. en zone paramètres.
: RELINK   (S adparms,base -- ) \ rétablit liens avec dict.
   SWAP  4 + ( adparm_Lfa )  #THREADS @  DO
     2DUP \ pointe un &Lfa en zone param.; si 0, suivant.
     I 2* + @  ( &Lfa) ?DUP IF + ( &Lfa+base=Lfa )
     CURRENT @  I 2* +  ( brin_i )
     DUP @ -ROT !        \ place nouvel LFA ds CONTEXT.
     OVER  #THREADS 2* +  I 2* +  @ ?DUP ( &Lfa_inf )
 IF 3 PICK +  !          \ place l'ancien LFA ds le segment
ELSE DROP THEN           \ conditionnelle 'Lfa_sup'.
   ELSE DROP THEN        \ conditionnelle 'Lfa_inf'.
  LOOP 2DROP  ;
```

écran #6.

```
\ LODBIN : VALIDE?, (LBIN).                      02Jui88JaD

: VALIDE?  (S adparms -- ) \ fich. converti par SAVBIN ?
  20 + @  ['] TAB_BIT <>
  IF  CR BRILL ." Mauvais fichier d'overlay: "
   FCB1  .FILE  NORM CR CR  ABORT  THEN  ;

: (LBIN)          (S adr,fcb -- )
  DUP  OPEN-BIN          \ ouvre le fichier en lecture
  OVER  READ-BIN         \ lit le fichier, le place en 'adr'
  DUP   VALIDE?          \ 'overlay' OK?
  PARMS                  \ paramètres de chargement
  DUP  ALLOT             \ déplace le pointeur de dictionnaire
  RELOC                  \ calcule les nouvelles adresses
  RELINK  ;              \ et les nouveaux liens.
```

écran #7.

```
\ LODBIN : ("LBIN).                              02Jui88JaD

: ("LBIN)    (S adr-chaîne -- )
  FCB1  DUP  B/FCB ERASE  1+ SWAP CMOVE   \ de fichier valide.
  HERE  FCB1 (LBIN) ;
     \ la chaîne doit contenir un nom de fichier binaire.
\S
("LBIN) permet de créer des mots d'appel de programmes binaires.

Exemple : appel et exécution d'un éditeur

: EDIT [ DOS ] " EDITEUR BIN"  ("LBIN)
  CONTEXT @ HERE #THREADS 2* CMOVE
  HERE #THREADS LARGEST  LINK> EXECUTE ;

Voir aussi écrans #9 et 10.
```

écran #8.

```
\ LODBIN : LBIN.                                 02Jui88JaD

FORTH DEFINITIONS

: LBIN        \ LBIN <nom_fichier>  ( extension .BIN implicite )
  [ DOS ]
  FCB1 B/FCB ERASE
  FICH>FCB
  HERE FCB1 (LBIN)
  FCB1 CR CR BRILL  .FILE  ." chargé."  NORM CR CR
  [ FORTH  ] ;
```

écran #9.

```
\ Exemple: DECOMPILATEUR.                        02Jui88JaD
: DECOMPILATEUR  \ un fich. binaire DECOMPIL.BIN a été créé.
  [ DOS ] " DECOMPILBIN"  ("LBIN)
  CR CR BRILL ." Le décompilateur est chargé: "  CR CR
 ." Utilisation: "          CR
 ."      1 - pour décompiler un seul mot :" CR
 ."                  VOIR <mot> ou  V <mot>"  CR
 ."      2 - pour décompiler un groupe de mots :"  CR
 ."                  .MOTS"  CR
 ."      3 - pour les imprimer :"  CR
 ."                  PR-MOTS "  CR  CR
 ." Pour faire un fichier des mots du vocabulaire CONTEXT:"
  CR CR ."          SAUVE [d:]<nom_fich.ext> "
  CR CR ." Pour supprimer le vocabulaire DECOMP, faire :"
  CR    ."           DECOMPEND <Enter>"
  CR NORM ;
```

écran #10.

```
\ Exemple: SAVBIN                                02Jui88JaD

: SAVBIN    \ un fich. binaire SAVBIN.BIN existe sur disque.
   [ DOS ]  " SAVBIN  BIN "  ("LBIN)
  CR BRILL  ."   SAVBIN est chargé."
  CR CR     ."    Pour l'utiliser, ouvrir le fichier-source:
  CR CR     ."         OPEN <nom_fich-src.BLK>
  CR CR     ." puis faire:
  CR CR     ."       SBIN <marque> <nom_fich-bin>"
  CR CR ."  <marque> étant le premier mot du programme,"
  CR    ."  suivi du nom du fichier binaire sans extension."
  CR CR ."  Ce fichier sera créé sur le lecteur 'défaut',"
 CR ." sauf spécification contraire dans le nom du fichier,"
  CR ." avec une extension .BIN."
 CR CR ." Pour supprimer SAVBIN, faire SAVEND <Enter>"
 CR CR NORM  ;
```

écran #11.

***** SAVBIN *****

Ce programme permet la construction et la sauvegarde sur disque d'un fichier binaire, c-à-d directement chargeable dans le dictionnaire FORTH.

Il utilise un fichier source écrit selon la syntaxe Forth, le compile 2 fois dans le dictionnaire, sauve l'une des versions avec les paramètres nécessaires au rechargement.

Celui-ci est effectué par le programme LODBIN.

Le source utilisé doit être adapté à cette utilisation:
- le premier mot doit être connu de l'utilisateur, ce peut être un mot de marquage (MARK <mot>) ou un mot 'vide';
- seules les définitions sont transférées à partir d'un fichier binaire; les mots exécutables du source ne seront pas interprétées lors du transfert (vectorisations p. ex.)
- cette version suppose que l'écran #1 du fichier source est un écran de chargement (utilisation du mot OK dans LOC).

écran #12.

```
\ SAVBIN: chargement.                                  02Jui88JaD
ONLY  FORTH ALSO  DEFINITIONS
: *SAVBIN* ;    \ marque pour la version binaire.
: SAVEND ['] *SAVBIN* >VIEW (FORGET) ; \ pour l'effacer.
DOS ALSO DEFINITIONS
  1 8 +THRU
ONLY  FORTH ALSO DEFINITIONS
\S    CR CR  BRILL \ texte à placer dans le mot d'appel
 .( SAVBIN est chargé.) CR CR
 .( Pour l'utiliser, ouvrir le fichier-source:) CR CR
 .(     OPEN <nom fich-src.BLK> )  CR CR
 .( puis faire:)  CR CR
 .(      SBIN <marque> <nom_fich> )  CR CR
 .(  où  <marque>  sera le premier mot sauvegardé,) CR
 .(        <nom_fich> sera le nom du fichier binaire créé, avec
extension .BIN.)    CR CR NORM
```

écran #13.

```
\ SAVBIN : table.                                      02Jui88JaD
CODE 2**N (S n -- 2 puissance n )
  CX POP  1 # AX MOV  AX CL SHL  \ 1PUSH  C; remplacé par :
  AX PUSH  BX LODS  AX BX MOV  0 [BX] JMP C;
: BIT_COMPL    (S offset -- )  \ complémente un bit.
  8 /MOD >R 2**N R>           \ calcule l'adr de l'octet,
  TAB_BIT @ + CTOGGLE ;       \ et met à 1 le 'r'ième bit.
: LGR_TAB      (S lgr -- taille ) 8 / 1+ ;
: TAILLE_MAX   (S lgr -- lgr-tot : de l'image créée )
   DUP  LGR_TAB  +  B/REC  2DUP MOD - + ;
: CLR_TAB      (S adr,lgr -- )
   SWAP  TAB_BIT !  LGR_TAB  B/REC +  TAB_BIT @ SWAP ERASE  ;
: CONST_TAB    (S base=adr_début,lgr -- )
  2DUP  2* +  OVER  CLR_TAB
  2DUP + SWAP  0  DO   2DUP  I + C@  SWAP  I + C@
  <>  IF  I  BIT_COMPL  THEN  LOOP  2DROP ;
```

écran #14

```
\ SAVBIN : borne inférieure de sauvegarde.      02Jui88JaD
VARIABLE TAMP 30 ALLOT  TAMP 32 ERASE
: SAV_CONT  CONTEXT @ TAMP #THREADS 2* CMOVE ;
  \ sauvegarde CONTEXT ds TAMP, adr des LFA supérieurs.
: BORNE_LFA   (S cfa-inf --- )
  SAV_CONT   >LINK            \ le LFA le plus bas est une borne.
BEGIN
 TAMP #THREADS LARGEST   DUP  3 PICK  U>=
WHILE
 @ SWAP !
REPEAT  2DROP  DROP  ;

\S BORNE_LFA utilise les 4 LFA contenus dans CONTEXT (i.e. les
adr supérieures des 4 brins du vocabulaire) pour chercher les
4 LFA inférieurs du segment de dictionnaire à sauvegarder.
Si le segment n'en contient pas, TAMP est inchangé.
```

écran #15.

```
\ SAVBIN : recherche des liens inférieurs.          01Jui88JaD
: LIENS_INF (S cfa-inf ) \ calcule les 4 LFA inférieurs.
    BORNE_LFA
  TAMP #THREADS 0 DO  0 PRIOR !  \ drapeau.
   DUP I 2* + DUP @      ( tamp0 tampI tampI Lfa-Ti )
   CONTEXT @ I 2* + @    ( Lfa-i )
BEGIN
  2DUP U<                \ Lfa-Ti < Lfa-i ?
WHILE                    \ oui: Lfa suivant du brin i.
  DUP  PRIOR !  @        \ PRIOR conserve LFA précédent
REPEAT                   \ non: on efface, le précédent est
 2DROP  PRIOR @  SWAP  !     \ placé dans TAMP, un 0 sinon.
  LOOP  DROP  ;

\S Un 0 dans TAMP indique que le brin concerné n'a pas de re
présentant dans le segment de vocabulaire à suvegarder.
```

écran #16.

```
\ SAVBIN : sauvegarde des liens.                      01Jui88JaD
: SAV_LFA        (S cfa_inf,adr_sav,lfa_sup -- adr_sav+12 )
  #THREADS 0 DO        ( adparam+,lfa_sup)  \ les liens
  2DUP  CONTEXT @ I 2* + @    ( ... Lfa_i, i=0-3)
  SWAP  -                     ( Lfa_i-Lfa_sup )
   DUP 0< IF DROP 0 THEN    \ place un 0 dans TAMP si δ<0
  SWAP  I 2* +  !             ( δLfa,adparam_i )
  LOOP  SWAP #THREADS 2* +  2DUP + OVER ERASE +
        SWAP  DECAL @ -      \ calcul du Lfa_inférieur.
  TAMP  #THREADS 0 DO      ( ad-sav,lfa_inf,Tamp )
   3DUP  I 2* +  @    ( Lfa_Ti )  ?DUP
     IF SWAP - ( δLfa ) SWAP I 2* + ! \ δLfa ds zone param.
    ELSE  2DROP  THEN        \ si 0, inchangé.
  LOOP
 2DROP  #THREADS 2* +  ; \ pointe fin actuelle zone param.
```

écran #17.

```
\ SAVBIN : LOC.                                       01Jui88JaD
\ Localise les CFA du mot <nom> dans les 2 versions
\ compilées.
: LOC          \ LOC <nom>  (S -- cfa_inf,cfa_sup )
   OK          \ compile une 1ère fois le fichier COURANT.
  BL WORD  DUP  TAMP 32 CMOVE  FIND    ( cfa_inf )
   CR INV 0=  ABORT"  La marque n'a pas été trouvée" NORM
   WARNING OFF  OK  WARNING ON      \ 2ème compilation.
  TAMP  FIND                        ( cfa_sup )
   CR INV ABORT" Il me faut 2 copies ..." NORM
  2DUP - 255 AND             \ lgr divisible par 256 ?
   CR INV ABORT" Ajoutez UN octet au programme ..." NORM
  OVER  LIENS_INF   ;
\S Des erreurs de 'relogement' se produiront si la lgr est
multiple de 256 (100 hexa): il faut dans ce cas rajouter de
1 à 255 octets au programme source (xx ALLOT p. ex.).
```

écran #18.

```
\ SAVBIN : WRITE-PARMS.                              02Jui88JaD

: WRITE-PARMS          (S cfa_inf,vfld,lgr  -- )
  2DUP   TAILLE_MAX  +    ( cfa_inf,vfld,lgr,ad_parms)
   DUP  B/REC ERASE            \ efface zone paramètres.
  DUP  >R  !                   \ sauve lgr du segment de dict.,
  DUP  R@  2+  !               \ l'adr de compilation origine .
  R> 4 + SWAP          ( ad_parms+4,lfa_sup )
   SAV LFA                     \ les liens du segment,
  ['] TAB_BIT  SWAP  !  ;      \  valeur de contrôle

\S WRITE-PARMS place les paramètres du rechargement
(longueur, liens supérieurs et inférieurs, ... ) dans la zone
réservée au-dessus de la table de 'relogement'.
 Le tout sera sauvé sur disque.
```

écran #19.

```
\ SAVBIN : crée, écrit, ferme un fichier.          01Jui88JaD

: MAKE-BIN           (S fcb -- ) \ crée l'entête
  22  BDOS  DOS-ERR?           \ d'un fichier sur disque.
  ABORT" Pas de place dans le répertoire." ;

: WRITE-BIN          (S adr_déb,lgr,fcb -- )
  -ROT B/REC  /  1+          \ nbr de secteurs à écrire.
    0 DO  ( fcb adr )
  DUP  26  BDOS  DROP      \ place la DTA.
  OVER 21  BDOS  DROP      \ écrit un secteur.
  B/REC  +                 \ adresse du secteur suivant.
  LOOP  2DROP ;

: CLOSE-BIN          (S fcb -- )
  16  BDOS  DOS-ERR?  ABORT" Fermeture impossible" ;
```

écran #20.

```
\ SAVBIN : SBIN, mot de haut niveau.                  01Jui88JaD
 FORTH DEFINITIONS
: SBIN    [ DOS ]              \ SBIN <1er_mot> <nom_fichier>
  LOC  ( cfa_inf,sup)          \ adr extr. du segm. de dict.
  2DUP  SWAP  -  DECAL  !      \ sa lgr en octets.
  FICH>FCB                     \ nom du fich. -> FCB,
  FCB1  MAKE-BIN               \ et le crée.
  SWAP  >VIEW ( cfa_sup,base_inf)      \ adr basse du segm.
  DECAL @  CONST_TAB        \  crée table de 'relogement'.
  >VIEW  DECAL @  ( vfa_sup,lgr )
  2DUP  WRITE-PARMS            \ table des paramètres.
    TAILLE_MAX                 \ nbr d'oct. à écrire.
  FCB1  WRITE-BIN              \ écrit le fich. image
  FCB1  CLOSE-BIN              \ ferme le fichier
  CR  FCB1 BRILL .FILE  ."  écrit sur disque." NORM CR
   ;
```

Voici un développement Modula destiné à interfacer dans ce langage le générateur d'écran HISCREEN. Moi-même, j'utilise la version US bien moins chère.
Il n'y a pas en effet d'équivalent de write pour envoyer la chaîne de commande au module d'interception résident de HISCREEN. On crée donc une procédure HiCom (Commande: ARRAY OF CHAR).

```
(*
Title:    Definition module HiScreen
LastEdit: 20/06/88
Author:   Daneluzzo
System:   LOGITECH MODULA-2/86
*)

DEFINITION MODULE HiScreen;

EXPORT QUALIFIED
    HiCom;  (* Procedures *)

PROCEDURE HiCom (Commande: ARRAY OF CHAR);
  (* Commande contient la chaîne à émettre dans laquelle se
trouvent les ordres à envoyer au module résident de Hiscreen*)

END HiScreen.


(*
Title:    Implementation module Hiscreen
LastEdit: 20/06/88
Author:   Daneluzzo
System:   LOGITECH MODULA-2/86
*)

IMPLEMENTATION MODULE HiScreen;

FROM Strings IMPORT Length;
FROM SYSTEM IMPORT ADR,DOSCALL;
FROM Exec IMPORT Run;

PROCEDURE Raz (VAR Chaine: ARRAY OF CHAR); (* Vide une chaîne de
50 octets *)

VAR i: INTEGER;

BEGIN

  FOR i:=0 TO 49 DO
    Chaine[i]:=CHR(0);
  END;

END Raz;

PROCEDURE Lie (s1: ARRAY OF CHAR; VAR s2: ARRAY OF CHAR);
  (* Concatène s1 à s2 dans s2 *)

VAR i,j: INTEGER;

BEGIN
```

```
  i:=0;
  j:=0;
  WHILE s2[j]<>CHR(0) DO     (* On cherche l'indice du premier
blanc dans s2*)
     j:=j+1;
  END;
  WHILE (i<VAL(INTEGER,Length(s1))) DO    (* Puis s2 se remplit
par s1 *)
     s2[j]:=s1[i];
     i:=i+1;
     j:=j+1;
  END;

END Lie;

PROCEDURE Lance (Commande: ARRAY OF CHAR);
  (* Envoie Commande dans la routine DOS "Emmission d'un message"
*)

BEGIN

  DOSCALL(9h,ADR(Commande));

END Lance;

PROCEDURE HiCom (ch1: ARRAY OF CHAR);
  (* Prépare la commande Hiscreen complète *)

VAR Beg: ARRAY [0..2] OF CHAR;    (* Marqueur de début *)
    HiFonct: ARRAY [0..49] OF CHAR; (* Phrase de commande
Hiscreen *)

BEGIN

  Beg[0]:=CHR(19);   (* Marqueur Hiscreen *)
  Beg[1]:=CHR(255);  (* Marqueur Hiscreen *)
  Beg[2]:=CHR(1);    (* Marqueur Hiscreen *)
  HiFonct[0]:=' ';   (* Init chaîne *)

  Raz(HiFonct);      (* Met à blanc la chaîne *)
  Lie(Beg,HiFonct);  (* Place les marqueurs en début de la
commande *)
  Lie(ch1,HiFonct);  (* Met notre ordre dans la commande *)
  HiFonct[48]:=CHR(1);  (* Marqueur de fin Hiscreen *)
  HiFonct[49]:='$';   (* Fin de chaîne interruption 9h *)
  Lance(HiFonct);    (* Envoie la commande au module résident *)

END HiCom;

END HiScreen.
```

Voici un petit exemple d'utilisation de la procédure HiScreen:

```
(*
Title:    Essais modules HiScreen
```

```
LastEdit: 19/08/88
Author:   Daneluzzo
System:   LOGITECH MODULA-2/86
*)

MODULE HiUse2;

FROM SYSTEM IMPORT ADR,DOSCALL;
FROM Exec IMPORT Run;
FROM Terminal IMPORT WriteString,ReadString;
FROM InOut IMPORT ReadInt;
FROM HiScreen IMPORT HiCom;

VAR Ok: BOOLEAN;
    RC: INTEGER;

BEGIN

  Run('C:\CDES\PROG\BEG.COM','',Ok);
  Run('C:\CDES\PROG\DISPLAY.COM','',Ok);
  HiCom('USE,C:\TEXTE\HS\ECRANS\LIV-01.AID,RC');
  ReadInt(RC);
  IF RC<>0 THEN WriteString("Ecran LIV-01 absent") END;

  Run('C:\CDES\PROG\END.COM','',Ok);

END HiUse2.
```

Suite de la page 16

implementation model, Laxen and Perry's F83. This code should be easily adaptable to any Forth-83 system, however, and probably to many other Forths as well. On screen 67 we give the definitions of most of the non-standard F83 words we use, in case you need them. Unfortunately, there are several other words for which we can't do this without getting bogged down in a mass of irrelevant detail. But we will now try to give enough information for our purposes here.

Firstly, the words `DEFER` and `IS`, respectively, create and redirect an execution vector. These words have been fully described by Henry Laxen in two "Techniques Tutorials" (*FD* III/6, V/6) that are well worth reading. Secondly, a couple of non-standard words are called from `FIND`, which we redefine on screen 72. However, the change we make to the definition of this word is very minor, and comes right at the end. Thus, we can ignore the details of the workings of `FIND`. In fact, even if you don't have F83, the change to `FIND` will probably be exactly the same. Thirdly, we need to refer here to the F83 word `HEADER`. The change is simply to rename it `(HEADER)`, so again we don't need to go into the details of its workings. We have more to say about it below.

Our scheme operates by separating the headers of the internal words. These go into a special area, the separated heads area (SH, for short) and are later forgotten. This is not as simple as it sounds, since internal words will have external words compiled after them, at which time the internal names must still exist. Later, if we forget the internal names by simply chopping the dictionary, we will lose the external names, too. Rather, we must follow the dictionary links and unlink any internal names, leaving the rest of the dictionary intact. We provide here a new forgetting word to do this — the regular `FORGET` need not be changed.

Now we will go into a bit more detail. The SH area has its own "dictionary pointer," the variable `SH-DP`. The base and top of the area are pointed to by two other variables, `SH-DP0` and `SH-TOP`. I am defining these as system variables here, although you may want to make them user variables if you do any multitasking on your system.

Each header in the SH area looks exactly like a normal header. The difference is that, instead of being followed by a code field, it is followed by a pointer to the code field. The code field itself sits in its usual position in the dictionary (and, of course, has no header in front of it). Here is how we set up this kind of header. In F83, headers are laid down by the word `HEADER`, which returns with the dictionary pointer `DP` pointing to where the code field will go. `CREATE` calls `HEADER`, then stores the code field. Our modification to `HEADER` is simply to make it vectored. So rename `HEADER` to `(HEADER)` and follow it with the code:

```
DEFER HEADER'
(HEADER) IS HEADER
```

When internal words are being compiler, `HEADER` is redirected to execute `SEP-HEADER`, which lays down a separated header and pointer to the code field. As required, we leave `DP` pointing to where the code field will go. `SEP-HEADER` is defined on screen 67. Notice that this word calls `(HEADER)` to do most of the work.

The F83 word `NAME>` takes the address of a name field and returns the corresponding compilation address (in F83 and most Forths, this is the same as the address of the code field). This word must be redefined so that, if the name is in the SH area, the pointer following the name is used to obtain the compilation address. This we do on screen 66. Other Forths should use either this word or something very similar. Any other words that go from a field within a header to the corresponding code field or parameter field will also need to be checked. If they use `NAME>` (or its equivalent) to make the transition, there's no problem. In F83, the only word in this category that doesn't use `NAME>` is `FIND`. As we mentioned above, the necessary change to this word is very minor. For completeness, we give the whole definition on screen 72.

This brings us to the new forgetting word. One of its features is that it takes a pair of addresses and forgets everything between them. So we will call it `<FORGET>` (pronounced "forget between"). Its definition is given on screen 68. Note the use it makes of the two other words on that screen, `UNLINK` and `TRIM`. `TRIM`, in its turn, uses `UNLINK` — it is `UNLINK` that does all the hard work.

F83 has a multi-thread dictionary structure, with the constant `#THREADS` defining the number of threads. `TRIM` contains a `DO` loop over the threads, calling `UNLINK` for each one. If your system does not have a multi-thread dictionary, removing the `DO` loop from `TRIM` may be sufficient.

`<FORGET>` can take a few seconds to execute, depending on your processor. This may not matter, as it won't be used with high frequency. However, if you can rewrite `UNLINK` in code for your particular processor, so much the better. This may save a few bytes of memory as well.

If you want to, you can replace the regular F83 forgetting scheme with this new one. (Thus, our `TRIM` has the same name as its F83 counterpart.) Simply redefine

```
: (FORGET)   DUP -1
 <FORGET>   DP !  ;
```

and replace `TRIM` with our new one. `FORGET` itself need not be altered. The advantage of doing this replacement is to save space, since our forgetting scheme is really an extension of the old one and covers much of the same ground. It works identically if there are no SH names, and the high limit is set to the highest address. If `UNLINK` is written in code, `FORGET` should run as fast as before.

The word `SET_SH_AREA` (screen 67) sets up the SH area by initializing the three pointers to this area. As written, it sets up 2000 bytes for the area in high memory, and 200 bytes below the current top of the parameter stack (which grows downwards). If this is not suitable for your system, change it as necessary. The number 2000, set up as the constant `SH_AREA_SIZE`, I found to be sufficient for my needs. If you have a big application, you may need to increase this number. The variable `SH-MAX` keeps track of the maximum size required so far for the SH area, so you can find by experimentation what size you need. Separate modules (i.e., not nested) reuse the same space in the SH area, to minimize the required size.

Modules may be nested in the fashion:

```
INTERNAL ...
INTERNAL...EXTERNAL...
 MODULE
EXTERNAL ...
MODULE
```

This is the same as provided in Dewey's original implementation of these words. Notice that once we are `EXTERNAL`, we can't become `INTERNAL` again in the same module. This could, however, be implemented. Being such perceptive readers, you will no doubt have noticed that `<FORGET>` is more general than we really need. Internal names to be forgotten will always occur in a single, contiguous cluster, whereas our implementation of `<FORGET>` will allow them to occur randomly, anywhere in the dictionary search order. This was, in fact, easier and shorter to implement (although slower). But it does mean that we can implement the more complicated module format, if necessary, without much trouble. I haven't done it here, feeling there may well be different schools of thought as to whether this feature will be desireable. Code is probably easier to follow if all `EXTERNAL` definitions are together, and this can usually be arranged quite easily. But go ahead and implement the more complicated format if you want to.

Our scheme for the temporary loading of entire modules (screen 71) is a straightforward application of `<FORGET>`. The word `TEMP_MODULE` saves `DP` on the stack, then resets it to halfway between where it was and the bottom of the SH area. You then load the temporary module, and use `END_TEMP` to set `DP` back to where it was. Hopefully, there is enough room between there and the temporary module for the definitions you

now want to load. When this is done, **FORGET_TEMP** forgets the temporary module. If the temporary module has defined any vocabularies, remember to remove them from the search order before **FORGET_TEMP** — unless you find crashes entertaining! A very simple example is shown on screen 75.

Michael Hore is a missionary Bible translator with a programming background. He works among a remote group of Aborigines and uses an LSI 11/2. He is progressively moving all his software over to Forth, "...the way computers were meant to be programmed."

.../...

```
65

\ Separate header area                                    May85 MRH

      VARIABLE  SH-DP0
      VARIABLE  SH-DP
      VARIABLE  SH-TOP
 2000 CONSTANT  SH_AREA_SIZE
      VARIABLE  EXTNL?
      VARIABLE  SH-MAX




: SH-HERE  ( -- addr )  SH-DP @  ;
```

```
1065

\ Separate header area                                    May85 MRH
SH-DP0   points to the base of the SH area.
SH-DP    Current SH area pointer.
SH-TOP   Points to the top of the SH area.
   Note: these three are all zero if the SH area has not been
   initialized.
SH_AREA_SIZE  Size we allocate for SH area.  Alter if necessary.
EXTNL?   A variable to allow us to check that the order
   EXTERNAL...MODULE is always followed (otherwise we'd crash).
SH-MAX        Records the maximum SH space used so far.



SH-HERE  Fetches the current SH area pointer.
```

```
66

\ SH area, cont.                                          May85 MRH
: WITHIN?  ( n lo hi -- n f )  OVER - >R  OVER SWAP -
    R> SWAP U< NOT   ;


: SEP_HDR?  ( addr -- addr f )  SH-DP0 @  SH-TOP @  WITHIN?  ;

: ?SEP_HDR>  ( ?acf -- acf )  SEP_HDR?  IF  @  THEN   ;



: NAME>  ( anf -- acf )
    1 TRAVERSE 1+  ?SEP_HDR>   ;
```

```
1066

\ SH area, cont.                                          May85 MRH
WITHIN?  Is n within the range lo to hi inclusive?  My version
   of this word has a couple of peculiarities - the test value
   is not popped, and the arithmetic is unsigned and circular,
   wrapping around from 64K to 0.
SEP_HDR?   Is addr within the SH area?

?SEP_HDR>  ?acf is either the address of a code field (acf),
   or a pointer to one if we are in the SH area.  Returns the
   acf.

NAME>   Converts the addr of a name field (anf) to an acf.
   TRAVERSE (F83) is the same as in Fig-FORTH.  Note that in F83
   link fields precede name fields, so that after the TRAVERSE
   we are normally looking at the code field.  At this point we
   insert ?SEP_HDR>.  This is the only change to the definition.
```

```
67

\ SH area, cont.                                          May85 MRH

: SEP_HEADER ( --(name) )  HERE ( save )  SH-HERE DP !
    (HEADER)  ( saved dp )  DUP ,   DP @  SH-DP !  DP !   ;

: SET_SH_AREA   SP@ 200 -  DUP  SH_AREA_SIZE -
    DUP  HERE  200 +  U<  ABORT" Not enough room in memory"
    DUP  SH-DP0 !  SH-DP !  SH-TOP !   ;

: ?SET_SH_AREA   SH-DP0 @ 0=  IF  SET_SH_AREA  THEN   ;

\ 0  CONSTANT  FALSE
\ -1 CONSTANT  TRUE
\ : ON   ( addr -- )  TRUE  SWAP !   ;
\ : OFF  ( addr -- )  FALSE SWAP !   ;
\ : ?PAIRS  ( n -- )  = NOT ABORT" Unbalanced structure"   ;
```

```
1067

\ SH area, cont.                                          May85 MRH

SEP_HEADER   Lays down a header in the SH area.

SET_SH_AREA   Sets up the SH area.  Alter this definition if
   necessary for your system.

?SET_SH_AREA  Sets up the SH area if not done already.

Here are some non-standard words included in F83.  Use these
definitions if you need to.
```

68

```
\ (FORGET), etc.                                         May85 MRH
: UNLINK  ( lo hi 1st-link -- lo hi 1st-link )  DUP >R
   BEGIN   ?DUP
   WHILE   DUP >R @
      BEGIN  2 PICK 2 PICK  WITHIN?  WHILE  @  REPEAT
      DUP  R> !
   REPEAT   R>   ;

: TRIM  ( lo hi voc-link-addr -- lo hi voc-link-addr )
   [ #THREADS 2* ] LITERAL  -
   #THREADS 0 DO   UNLINK  2+   LOOP   ;

: (FORGET)  ( lo hi -- )  OVER  FENCE @  U<  ABORT" Below FENCE"
   VOC-LINK  UNLINK
   BEGIN  @ ?DUP  WHILE  TRIM  REPEAT
   2DROP  ;
```

1068

\ (FORGET), etc. May85 MRH

UNLINK Goes down a linked list starting with 1st-link, unlinking all links located at addresses within the range lo to hi (inclusive). This word could profitably be put into code.

TRIM For the given vocabulary, removes all words whose link fields are located within the range lo to hi. #THREADS is a constant giving the number of dictionary threads.

(FORGET) Forgets all words whose link fields are located within the range lo to hi. DP is not changed. We first check that lo is not below where FENCE points, to guard against wiping out the system. Then we use UNLINK to remove any vocabularies that are to be forgotten. (VOC-LINK is the head of the vocabulary list.) Then we go down the pruned list of vocabularies and call TRIM for each one.

69

```
\ Modular programming words                              May85 MRH
: SAVE_HEADER  ( -- acf 20 )   ['] HEADER  >BODY @  20   ;

: INTERNAL    EXTNL? @
   ABORT" INTERNAL follows EXTERNAL - probably MODULE omitted"
   ?SET_SH_AREA   SH-DP @  ( save )   SAVE_HEADER
   200 SH-DP +!   ['] SEP_HEADER IS HEADER   ;

INTERNAL


: RESTORE_HEADER  ( old-hdr-acf 20 )  20 ?PAIRS   IS HEADER  ;

: (SH_FORGET)   ( faddr -- )
    DUP  SH-TOP @  (FORGET)   SH-DP !  ;
```

1069

\ Modular programming words May85 MRH

SAVE_HEADER Saves the current setting of HEADER (DEFERred).

INTERNAL Subsequent definitions will have separate headers. Note we reserve 200 bytes in the SH area for EXTERNAL names, in case this is a nested INTERNAL.

Now INTERNAL is defined, we can make use of it right away.

RESTORE_HEADER Restores the previous setting of HEADER.

(SH_FORGET) Forgets all (SH) names above the limit, faddr.

70

```
\ Modular programming words, cont.                       May85 MRH

: (EXTERNAL)  ( old-sh-dp old-hdr-acf 20 )    RESTORE_HEADER
    SH-HERE  SH-DP0 @  -  SH-MAX @  MAX  SH-MAX !
    SH-DP !   EXTNL? ON   ;

(EXTERNAL)

: EXTERNAL  (EXTERNAL)   ;



: MODULE   EXTNL? @ NOT
   ABORT" MODULE follows INTERNAL - EXTERNAL omitted"
   EXTNL? OFF  SH-HERE  (SH_FORGET)   ;
```

1070

\ Modular programming words, cont. May85 MRH

(EXTERNAL) As for EXTERNAL below, but won't be accessible later, as an INTERNAL is now current. Needs to be used to make us external again, so EXTERNAL itself will be accessible.

EXTERNAL Subsequent defined names go where they were going before the last INTERNAL. These names will still be accessible after MODULE.

MODULE Forgets all names defined between INTERNAL and EXTERNAL. Names after EXTERNAL are still accessible.

71

```
\ TEMP_MODULE, etc.                                      May85 MRH

: TEMP_MODULE    ?SET_SH_AREA   HERE ( save )   SAVE_HEADER
    ['] (HEADER) IS HEADER
    HERE  SH-DP0 @  OVER -  2/ [ HEX ] 7FFE AND [ DECIMAL ]
    +  DP !   ;


: END_TEMP   RESTORE_HEADER   DP !   ;


: FORGET_TEMP   HERE  SH-DP0 @   (FORGET)   ;



MODULE
```

1071

\ TEMP_MODULE, etc. May85 MRH

TEMP_MODULE Marks the start of a module (such as the assembler) which is to be forgotten in toto once it has finished. This word leaves DP pointing to where the module will be loaded. Currently this is half-way between HERE and the bottom of the SH area. You can change this if necessary.

END_TEMP Used after the temporary module is loaded. Restores DP to its usual position.

FORGET_TEMP Used when the temporary module is no longer needed. Forgets it using (FORGET), so nothing else is affected.

72

```
\ Modified FIND                                    May85 MRH

: FIND   ( addr -- acf flag | addr false )
   PRIOR OFF   FALSE   #VOCS 0
   DO   DROP CONTEXT I 2* + @ DUP
      IF   DUP PRIOR @ OVER PRIOR !   =
         IF  DROP FALSE
         ELSE  OVER SWAP HASH @
               (FIND)  DUP ?LEAVE
   THEN THEN   LOOP
   DUP  IF  SWAP  ?SEP_HDR)  SWAP  THEN   ;
```

1072

\ Modified FIND May85 MRH

The first part of this definition is copied straight from F83.

This extra line is the only change to the definition.
If the name was found, call ?SEP_HDR) to ensure we have the address of the code field.

75

```
\ Module example                                   Mar86 MRH

INTERNAL

: (CHANGE_CASE)  ( c -- c' )
   ASCII A  ASCII Z  WITHIN?  SWAP  ASCII a  ASCII z  WITHIN?
   ROT OR  IF  BL XOR  THEN  ;

EXTERNAL

: CHANGE_CASE  ( addr len -- )  BOUNDS
   DO  I C@ (CHANGE_CASE)  I C!  LOOP  ;

MODULE
```

1075

\ Module example Mar86 MRH

This screen gives an example of a very simple module, which provides a single word to the outside world.

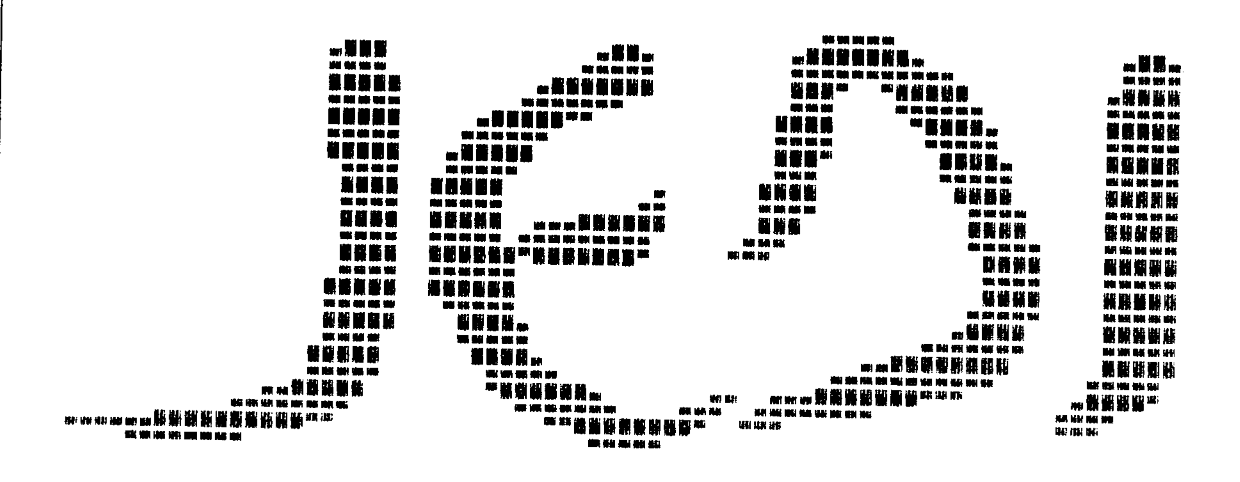
(CHANGE_CASE)
Changes the case of the given character, if it is alphabetic.

User word:

CHANGE_CASE
Changes the case of the string (addr, len).
BOUNDS (F83) is equivalent to OVER + SWAP .

Although CHANGE_CASE uses (CHANGE_CASE), the latter is now not accessible, and its name is not taking up any memory space.

CONNECT YOU TO THE FRENCH FORTH BBBS   Phone: (33) 36.43.15.15 Keyword: SAM*JEDI
PROTOCOL V23b (1200/75 bds - TELETEL via TRANSPAC) and...
"May be have the FORTH with you!"



Donnez-vous un nom